KURT SCHARNBACHER

LERNE BASIC MIT DEM COMMODORE 116/16/PLUS 4

Eine Anleitung zum Selbststudium in 10 Lektionen

1. Auflage 25000 Okt. 1985
2. Auflage 25000 Nov. 1985
3. Auflage 60000 Jan. 1986
4. Auflage 20000 Febr. 1986

Alle in diesem Handbuch gegebenen Informationen wurden überprüft und sind daher zuverlässig. Für eventuelle sachliche Fehler kann jedoch keinerlei Verantwortung übernommen werden.

Commodore Büromaschinen GmbH 1985
Artikel-Nr. 580117/1.86

LERNE BASIC MIT DEM COMMODORE 116/16/PLUS 4

Eine Anleitung zum Selbststudium in 10 Lektionen

Prof. Dr. Kurt Scharnbacher



Inhaltsverzeichnis

Vorwort

Computer und Computer-Systeme spielen in unserer heutigen Welt eine wichtige Rolle, ihre Bedeutung nimmt mehr und mehr zu. Aus diesem Grund ist es erforderlich, daß der Computer, seine Leistungsfähigkeit und seine Programmierung nicht als Geheimwaffe betrachtet werden, die nur wenige lernen können, sondern es erscheint wünschenswert, daß jeder etwas über die Möglichkeiten dieser Geräte weiß.

Der beste Weg zum Verständnis eines Computers besteht darin, die „Sprache des Computers" zu lernen. BASIC ist ohne Zweifel diejenige Computersprache, die am weitesten verbreitet und am leichtesten erlernbar ist. In dem vorliegenden Buch soll versucht werden, für Laien verständlich die Grundprinzipien dieser Programmiersprache zu erklären.

Leider ist BASIC noch nicht in allen Einzelheiten genormt, deshalb ist es notwendig, die eine oder andere Besonderheit aus dem Gerätehandbuch zu entnehmen. Das Buch ist so aufgebaut, daß es für das Selbststudium zum Erlernen von BASIC geeignet ist. Deshalb wird jedem Kapitel das Lernziel vorgeschaltet, ergänzt um die jeweiligen BASIC-Befehlsworte und die zugehörigen Struktogrammsymbole.

Als Demonstrationsbeispiel für die BASIC-Befehlsworte wurde ein einziges Beispiel, und zwar die Berechnung „Einfacher Bankzinsen" gewählt. Dieses Beispiel wird in jedem Kapitel erneut aufgegriffen und variiert, so daß der Leser sich nicht mehr mit dem Rechnungsproblem beschäftigen muß, sondern sich ganz der Programmierung widmen kann. Auf die Dateiverarbeitung wurde zugunsten der Logik verzichtet. Zur Selbstkontrolle dienen Kontrollfragen in jedem Kapitel, deren vollständige Lösungen am Ende des Buches stehen. Hier finden sich auch weitere kleine Beispiele, die der Leser selbst zu erarbeiten hat. Auf diese Weise soll Schritt für Schritt von der bloßen Kenntnis der Sprachelemente zur Programmierlogik geführt werden.

Hofheim-Wallau, im Juli 1985 Prof. Dr. Kurt Scharnbacher

1. Grundbegriffe der Datenverarbeitung

Die Computer bestehen aus mehreren Elementen, die man in zwei große Gruppen aufteilen kann.

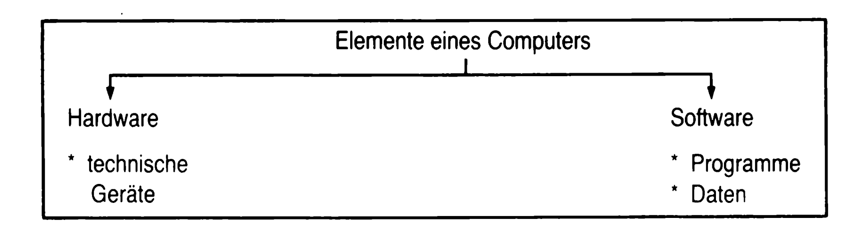


Abb. 1: Elemente eines Computers

1.1 Hardware

Lernziel: Der Leser soll verstehen, aus welchen technischen Teilen sich ein Computersystem zusammensetzt und welche Aufgaben sie innerhalb des Systems haben.

Sie haben einen neuen Computer COMMODORE 116/16/+4 gekauft und brauchen für die weitere Arbeit dazu einen Monitor (Fernsehgerät oder speziellen Datenmonitor), ein Diskettengerät (Floppy-Disk) oder Cassettengerät und eventuell einen Drucker. Diese Geräte sind Ihre **Hardware.**

Generell können um die Zentraleinheit (ZE) bzw. CPU (Central Processing Unit) verschiedene periphere Einheiten gruppiert sein:

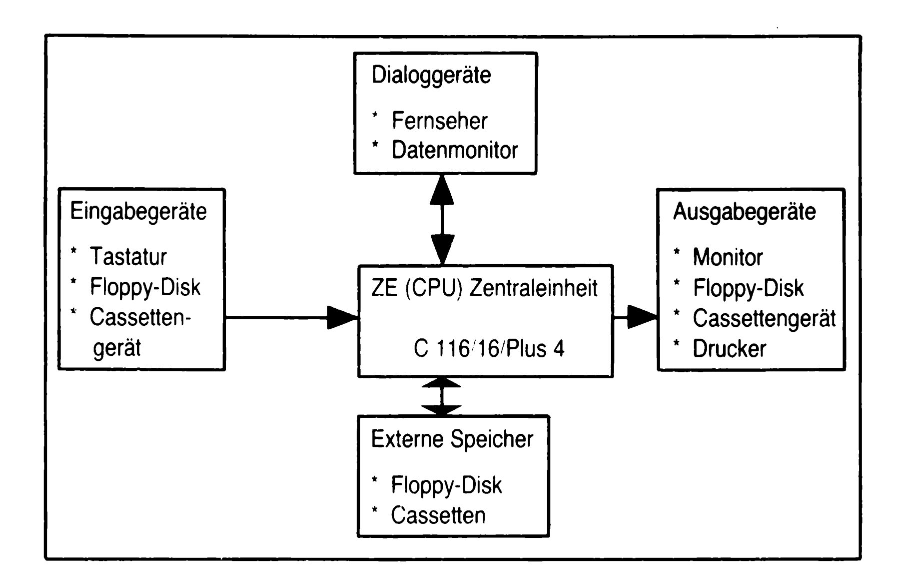


Abb. 2: Elemente der Hardware

Dabei haben die Geräte folgende Aufgaben:

Eingabegeräte — Sie dienen ausschließlich der Eingabe von Informationen in die ZE.

Ausgabegeräte — Geben die Informationen/Berechnungen der ZE aus.

Dialoggeräte — Sie zeigen an, welche Informationen in die ZE ein- bzw. über sie ausgegeben werden.

Externe Speicher — Sie übernehmen zusätzlich zur Ein- und Ausgabe von Informationen auch deren Speicherung.

Die Verarbeitung der Informationen im Computer können im Vergleich zum Menschen dargestellt werden. Die ZE ist „das Gehirn des Computers", und ist analog zum menschlichen Gehirn aufgebaut.

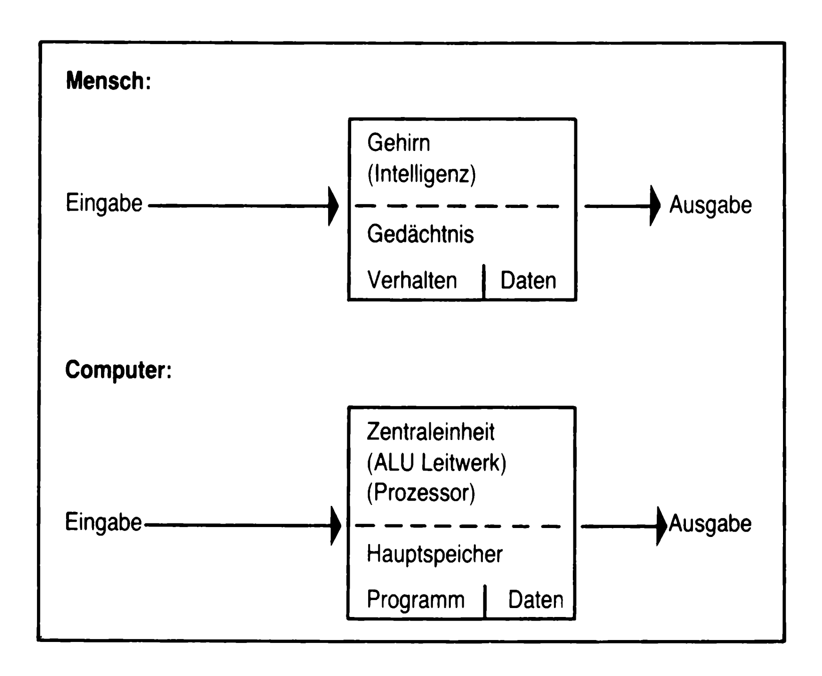


Abb. 3: Vergleich Mensch – Computer

Die Eingabe erfolgt beim Menschen durch Reize über Auge, Ohr, Nase u.a. und entspricht beim Computer der Eingabe z.B. über die Tastatur. Sie wird beim Menschen im Gehirn verarbeitet, im Computer in der ZE, die aus der Arithmetik Logical Unit (ALU) sowie dem gesamten Steuer- bzw. Leitwerk besteht. Die Ausgabe erfolgt beim Menschen wiederum durch Handlungen (Sprechen u.a.), beim Computer über die verschiedenen Ausgabeeinheiten.

Um den technischen Aufbau eines Computers sehen zu können, muß man die Abdeckung der ZE abnehmen. Darunter entdeckt man eine Fülle von elektronischen Bauelementen, die durch Drähte und Leitungen miteinander verbunden sind. Diese Bauelemente allein können keine für uns Menschen nützliche Leistung vollbringen. Einem Computer muß immer erst von dem Menschen „beigebracht" werden, was er tun soll, d.h. er muß programmiert werden. Die Gesamtheit der Programme, die bewirken, daß ein Computer nützliche Arbeit verrichtet, bezeichnet man als **Software.**

1.2 Die Software

Lernziel: Der Leser soll verstehen, welche Bedeutung die Software für den Ablauf innerhalb eines Computersystems hat.

Software ist Information und wird unterteilt in Programme und Daten. Wir wollen im folgenden speziell eingehen auf die Programme.

Beim Öffnen des Gehäuses eines Personalcomputers entdeckt man neben einem Netzteil, Verbindungsleitungen, verschiedenen Steckern als Schnittstellen zum Kontakt mit den Peripheriegeräten auch Platinen als Leiterplatten, auf denen Schaltkreise (Chips) montiert sind. Wichtig sind für uns hier die Chips, wobei ein Chip ein kleines Plättchen aus Silicium ist, das im Zuge der Herstellung bestimmter Schaltelemente zu einer Einheit integriert wurde. Einen Chip bezeichnet man deshalb auch als integrierten Schaltkreis (IC = Integrated Circuit). In diesen integrierten Schaltkreisen sind Programme aufgeschmolzen, die die Arbeit mit einem Computer erst ermöglichen.

In diesen Chips ist als Programm das Betriebssystem enthalten, das mit seinen Steuer-, Dienst- und Übersetzerprogrammen als Mittler zwischen den Anwenderprogrammen und dem Computerkern (Hardware) dient.

Damit ist ein solcher Chip zugleich ein Speicherbaustein, der der Speicherung von Programmen und Daten dient. Man unterscheidet zwei Speicherarten:

ROM (Read Only Memory) Als nur Lesespeicher kann der Benutzer nur lesen, was von den Ingenieuren der jeweiligen Herstellerfirma hier fest gespeichert wurde (als wichtigstes: das Betriebssystem, das die Arbeit mit dem Computer überhaupt ermöglicht).

RAM (Random Access Memory) Ein Schreib-Lese-Speicher, d.h. ein Direktzugriffspeicher, in dem der Benutzer seine Anwenderprogramme, d.h. die Programme, die er selbst geschrieben hat oder die Daten, die er benutzen möchte, speichern kann.

Damit sind in einem Micro-Computer folgende Teile enthalten:

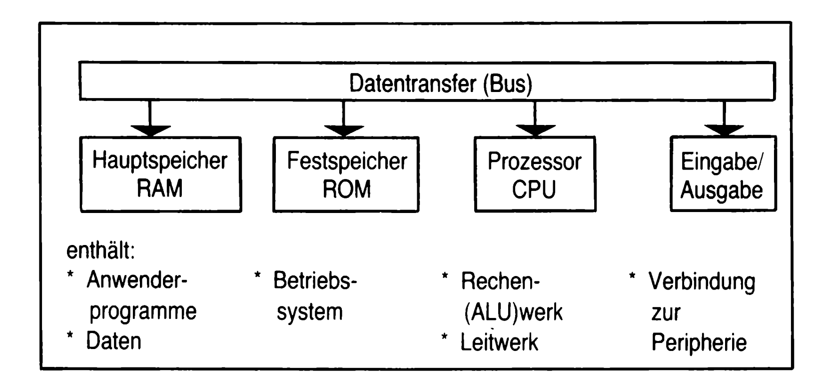


Abb. 4: Modell des Aufbaus eines Micro-Computers (Bausteine)

Jedes Programm läuft folgendermaßen ab:

Nach dem Start schickt der Micro-Computer über den Datentransfer die Adresse des Progammbefehls an den Speicher, in dem sich das Programm befindet. Der Speicher transportiert den unter dieser Adresse gefundenen Befehl über den Datentransfer an den Micro-Prozessor. Der Befehl wird ausgeführt und nach der Ausführung schickt dieser wiederum die Adresse des zweiten Programmbefehls an den Speicher usw.

Jedes Programm ist nach folgendem Prinzip zu erstellen:

Eingabe – Verarbeitung – Ausgabe.

Ein Programm, das abgearbeitet wurde, muß zu einem Ende kommen, d.h. die Endanweisung muß programmiert werden. Sie steht am Ende eines jeden Programms und hat ebenfalls eine Anweisungsnummer. Das BASIC-Schlüsselwort END beendigt die Ausführung eines mit dem Befehl RUN gestarteten Programmes.

Die Eingabe, Verarbeitung und Ausgabe bzw. auch andere Informationen können ausgedruckt werden, und zwar mit dem BASIC-Schlüsselwort REM. Diese REM-Anweisung ermöglicht es, in das Programm selbst Bemerkungen einzufügen, die nur bei dem Auflisten des Programmes erscheinen, nicht aber bei RUN. Damit beeinflussen Sie den Lauf des Programmes nicht.

Damit besteht die Software aus den vorprogrammierten Chips und den Anwenderprogrammen. Diese Anwenderprogramme können in unterschiedlichen Programm-Sprachen geschrieben werden.

1.3 Programmiersprachen

Lernziel: Der Leser soll verstehen, welche Bedeutung Programmiersprachen für Micro-Computer haben.

Das vom Anwender erstellte Programm, das ganz auf bestimmte Zwecke ausgerichtet ist, muß vom Computer verstanden werden. Deshalb benötigt der Computer ein Betriebsprogramm, das ihn befähigt, die im Anwenderprogramm stehenden Befehle auszuführen.

Der Computer selbst kann aber die über eine Tastatur eingegebenen Buchstaben bzw. Zahlen n i c h t verstehen. Sie müssen umgesetzt werden in maschinenverständliche Sprachelemente. Diese maschinenverständliche Sprachelemente sind, neben anderen, im sog. ASCII-Code (ASCII – American Standard Code for Information Interchange) genormt.

Dies geschieht durch die Übersetzung der Anwenderprogramme in die sogenannten Maschinenprogramme.

Die Übersetzung kann erfolgen durch:

Compiler übersetzt das geschriebene Anwenderprogramm in ein Gesamtprogramm in Maschinenspache, das anschließend insgesamt abgearbeitet werden kann.

Interpreter übersetzt während des Programmlaufs eine Anweisungszeile nach der anderen aus der Programmiersprache in die betreffenden Befehle der Maschinensprache.

Eine Programmiersprache unter vielen, in der Anwenderprogramme vom Besitzer eines Computers komfortabel geschrieben werden können, ist die Sprache BASIC (BASIC = Beginners, All Purpose, Symbolic, Instruction Code).

BASIC ist eine normierte Programmiersprache, die aber inzwischen zahlreiche Versionen erhalten hat. Für die Geräte Commodore 116/16/+4 gilt die BASIC-Version 3.5. Frühere Versionen der Firma Commodore waren BASIC 2.0 und BASIC 4.0.

Die BASIC-Version 4.0 ist die Weiterentwicklung des BASIC 2.0, das die Standardsprache des Commodore 64 ist; der wichtigste Unterschied zum BASIC 2.0 besteht darin, daß der Zugriff auf Disketten-Dateien sehr komfortabel ist.

1.4 Die Phasen der Programmerstellung (Anwenderprogrammierung)

Lernziel: Der Leser soll erkennen, daß ein Programm geplant und anschließend realisiert werden muß.

Grundsätzlich unterscheidet man bei der Programmierung die Phase der Planung (Problemanalyse, Lösungsalgorythmus, Programmlogik) und die Phase der Realisierung (Codierung, Implementierung, Test, Fehlerbeseitigung).

Dabei kann die Erstellung eines Programms nach folgenden Schritten erfolgen:

1. Schritt: Am Anfang steht die Problemstellung, in der Ziel und Zweck des Programms zu Formulieren ist.

2. Schritt: In der Problemanalyse muß die Aufgabenstellung untersucht werden, und die Probleme, die auftreten können, sollen hier gesucht werden.

3. Schritt: Der Lösungsalgorithmus muß einen Lösungsweg angeben.

4. Schritt: Der Lösungsalgorithmus ist eine Folge von logischen Arbeitsschritten aufzugliedern (er ist der eigentliche Kern der gesamten Programmerstellung).

5. Schritt: Mit der Codierung wird die Programmlogik in eine Programmiersprache (hier BASIC) übertragen.

6. Schritt: Mit der Implementierung wird das Programm in die Maschine eingegeben.

7. Schritt: Das Programm ist zu testen und Fehler sind zu beseitigen.

8. Schritt: Das ablauffähige Programm kann eingesetzt werden.

Ein Problem zu analysieren bedeutet, daß dieses in seine Bestandteile zerlegt wird. Dabei sollte von der Idee ausgegangen werden, daß man vom einfachen zum schwierigen geht. Meist ist es so, daß die Ausgabedaten einfacher sind, so daß man von der Ausgabeanalyse über die Eingabeanalyse zur Verarbeitungsanalyse kommen sollte.

Der wichtigste Punkt beim Programmieren ist die Programmlogik, wobei natürlich die anderen Schritte nicht unberücksichtigt bleiben dürfen. Um die Programmlogik zu veranschaulichen, bedient man sich im allgemeinen der Logikdiagramme, die die Programmstrukturen zum Ausdruck bringen.

1.5 Logikdiagramme

Logikdiagramme sind Hilfsmittel, um den Ablauf eines Programms mit Hilfe von Symbolen (Struktogramm) zu veranschaulichen. Dabei unterscheidet man die folgenden vier Strukturen:

- Folgestruktur
- Auswahlstruktur
- Wiederholungsstruktur
- Unterprogrammstruktur.

Mit diesen vier grundlegenden Programmstrukturen lassen sich alle denkbaren Programmabläufe konstruieren, denn es gilt:

* Programmablauf linear/gerade = Folgestruktur
* Programmablauf vorwärts verzweigend = Auswahlstruktur
* Programmablauf rückwärts verzweigend oder Schleife = Wiederholungsstruktur
* Programmablauf unterteilend = Unterprogrammstruktur (eine Sonderform der Auswahlstruktur)

Jeder, zur Problemlösung erforderliche Programmablauf, kann durch geeignetes Anordnen dieser vier Programmstrukturen konstruiert werden.

Folgestrukturen
Werden Arbeitsanweisungen an den Computern aneinandergereiht, so spricht man von einer Folgestruktur. Dabei wird eine Anweisung nach der anderen wie in einer Linie abgearbeitet.

Struktogramm:	Beispiel:
Anweisung 1	Ausgabe der Fragestellung
Anweisung 2	Eingabe der Antwort
Anweisung 3	Berechnung des Betrages durch Multiplikation mit dem Preis je Einheit

Beispiel-Dialog auf dem Monitor:

Wie groß war die Bestellmenge?

20

Betrag 600,– DM

Abb. 5: Folgestruktur

Auswahlstruktur:
Auswahlstrukturen dienen dazu, aus einer Vielzahl von vorgedachten Möglichkeiten bestimmte Fälle auszuwählen.

Struktogramm:

Anweisung 1	
Bedingung erfüllt?	
nein	ja
Anweisung 2	Anweisung 3

Beispiel:

Ausgabe – Fragestellung

Eingabe – Antwort ja/nein

Fall 3 – Rechnung netto

Fall 2 – Rechnung brutto

Beispiel-Dialog auf dem Monitor:

Soll Skonto berechnet werden?

ja

Nettobetrag ist 588.- DM

Abb. 6: Auswahlstruktur

Wiederholungsstruktur:
Grundsätzlich führen Wiederholungsstrukturen zu Programmschleifen, die mehrmals durchlaufen werden. Das Ende dieser Wiederholungsschleife kann dabei entweder durch ein bestimmtes Symbol, das über die Tastatur eingetippt wird, oder im Programm durch eine Zählerschleife, die einen bestimmten vorgegebenen Durchlauf beinhaltet.

Struktogramm:

Wiederhole bis Anzahl erfüllt
Anweisung

Beispiel:

Eingabe – Anzahl der Schleifendurchläufe

Ausgabe – Zahlen

Beispiel-Dialog auf dem Monitor:

Zähle bis 3

1

2

3

Abb. 7: Wiederholungsstruktur

1.6 Kontrollfragen

1. Welche Bedeutung haben Eingabe/Ausgabe und Dialoggeräte?
2. Erklären Sie sich die Begriffe: ROM, RAM und CPU!
3. Machen Sie sich die Programmierfolge Eingabe – Verarbeitung – Ausgabe klar.
4. Welchen Punkt halten Sie bei den 8 Schritten der Programmerstellung für am wichtigsten?
5. Erläutern Sie sich die Begriffe Folge-, Auswahl- und Wiederholungsstruktur!

Für Ihre Lösungen:

2. Grundlegende Elemente der Programmiersprache BASIC

Lernziel: Der Leser soll den Unterschied zwischen Systemkommando und BASIC-Anweisungen erkennen und die Cursorsteuerung einsetzen können.

Schalten Sie jetzt Ihren Rechner ein, denn es ist nicht ratsam, die praktische Arbeit erst dann aufzunehmen, wenn man Kenntnisse der Programmiersprache BASIC erworben hat. Es ist vielmehr sinnvoller, sofort mit dem Programmieren zu beginnen, selbst dann, wenn einige Einzelheiten nicht verstanden werden und eventuell einige unerwartete Schwierigkeiten eintreten. Ein Commodore Computer kann durch fehlerhafte Eingabe nicht beschädigt werden.

Wenn Sie Ihren Rechner mit Hilfe der Anleitung des Herstellers an einen Monitor oder Farbfernseher korrekt angeschlossen haben, so erscheint jetzt auf dem Bildschirm die folgende Nachricht:

```
COMMODORE BASIC V3.5 12277 BYTES FREE
READY
C
```

Damit zeigt das jeweilige System an, daß es betriebsbereit ist. Der Commodore 116 stellt Ihnen als Anwender für Ihre Arbeit 12.277 Bytes von 16.000 (16 K = 16 x 1.000) freier Arbeitsspeicher zur Verfügung.

(BYTE = Ein Byte besteht aus 6 bis 16 Bits, die im Binärcode die jeweiligen Informationen wie Buchstaben, Zahlen, Sonderzeichen u.a. enthalten).

2.1 BASIC-Sprachelemente

Die Bedienung des Computers erfolgt über eine Tastatur und den Monitor **(= Dialogverkehr).** Wir benutzen dazu Befehlsworte der normierten Sprache BASIC und spezielle Befehlsworte des jeweiligen Computerherstellers. Neben diesen BASIC-Befehlsworten (oder BASIC-Schlüsselworten) muß es noch andere Befehlssätze geben, die bestimmte, gewünschte Wirkungen im Computer erzielen. Im einzelnen sind dies:

- Kommandos und Anweisungen
- Funktionen
- Operatoren
- Satz- und Sonderzeichen.

Bei den **Kommandos** und **Anweisungen** handelt es sich um Befehlsworte (sie entstammen der englischen Sprache) und müssen meist durch weitere Angaben (Argumente) ergänzt werden. Diese Kommandos werden vom Computer im Direktbetrieb verstanden und sofort ausgeführt. Die Anweisungen dagegen sind zwar auch Kommandos, sie stehen aber immer innerhalb eines Programms (und haben damit eine Zeilennummer).

Unter **Funktionen** versteht man in BASIC eine Anweisung zur Lösung eines ganz bestimmten Teilproblemes, wie z.B. eine mathematische Funktion. Eine solche Funktion stellt einen Wert bereit, der dann in einem weiteren Rechengang verwendet werden kann. Neben dieser mathematischen Auslegung des Begriffs Funktion können auch bestimmte Ausführungen, die gewollt werden, als Funktion bezeichnet werden. Bei den Commodore-Systemen C116/C16/+4 finden sich im oberen Bereich der Tastatur Funktionstasten; sie sind durch den Großbuchstaben F und einer Nummer gekennzeichnet. Aber auch die Taste HELP und die beiden danebenstehenden Tasten sind Funktionstasten.

Rechenzeichen oder Vergleichszeichen bezeichnet man in der BASIC-Sprache als **Operatoren.** Die Rechenzeichen, z.B. + oder −, bewirken eine bestimmte Rechenoperation, die auszuführen ist. Die Vergleichsoperatoren bewirken einen Vergleich zwischen zwei Ausdrücken, ein derartiger Vergleich ist entweder „wahr" oder „falsch". Je nach dem Ergebnis eines derartigen Vergleichs kann das Programm in verschiedenen Zweigen weiterlaufen.

Von Hersteller zu Hersteller variiert leider die Bedeutung der **Satz- und Sonderzeichen.** Zu diesen Sonderzeichen gehört das Komma, das Semikolon, der Doppelpunkt aber auch die Zwischenraumtaste und andere Tasten. Es bleibt hier nichts anderes übrig, als sich im jeweiligen Bedienerhandbuch speziell zu informieren, welche Wirkungsweise diese Sondertasten haben. Beim System Commodore C116/C16/+4 ist die Wirkungsweise des Kommas, des Semikolons und des Doppelpunkts speziell bei der PRINT-Anweisung und der INPUT-Anweisung zu beachten.

2.2 Systemkommandos

Bei den Commodore C116/C16/+4-Geräten befindet sich im Betriebsprogramm fest verdrahtet der **BASIC-Interpreter.** Man unterscheidet bei den meisten BASIC-Versionen zwei Betriebsarten, und zwar den

- Direktbetrieb (Direct Mode)
- Programmbetrieb (Programming Mode).

In beiden Betriebsarten arbeitet der Computer im Dialogbetrieb, d.h. auf dem Monitor erscheinen die Zeichen, die über die Tastatur eingegeben und von der Zentraleinheit entgegengenommen werden.

Wurde ein Programm in BASIC geschrieben und in den Computer eingegeben, so wird es immer durch das Kommando RUN gestartet. Die drei Buchstaben RUN müssen dabei einzeln über die Tastatur eingetippt werden und erscheinen auf dem Monitor.

Diejenigen Nachrichten, die auf dem Monitor über die Tastatur erscheinen, befinden sich aber noch nicht in der Zentraleinheit. Dazu ist eine besondere Taste erforderlich:

RETURN

Dies ist hier die RETURN-Taste, die bewirkt, daß Nachrichten, von der Tastatur über den Monitor in die Zentraleinheit gelangen. Mit anderen Worten bedeutet dies, daß die RETURN-Taste das Abschicken der Informationen in die Zentraleinheit bewirkt.

Eine Nachricht über das Drücken der RETURN-Taste erscheint **nicht** auf dem Bildschirm.

Ein Systemkommando ist dadurch gekennzeichnet, daß es grundsätzlich keine Anweisungsnummer hat.

2.3 Struktur einer BASIC-Anweisung

Eine BASIC-Anweisung besteht aus drei Teilen:

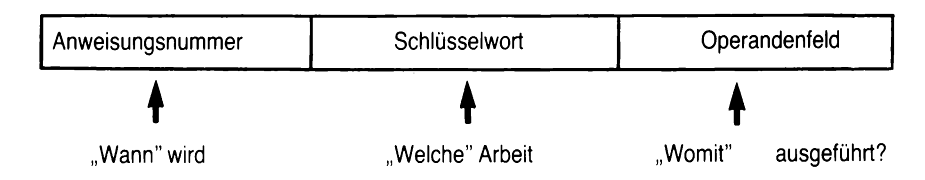


Abb. 8: BASIC-Anweisung

Ein BASIC-Befehl besteht grundsätzlich aus der Anweisungsnummer, dem BASIC-Schlüsselwort und dem Operandenfeld.

In Ausnahmefällen kann der eine oder andere Teil fehlen. Fehlt die Anweisungsnummer, so handelt es sich nicht um einen BASIC-Befehl, sondern um ein Systemkommando.

Die Bearbeitung der Programmbefehle durch den Computer fängt immer mit derjenigen Programmzeile an, die die niedrigste Zeilennummer hat. Danach wird Zeile für Zeile abgearbeitet.

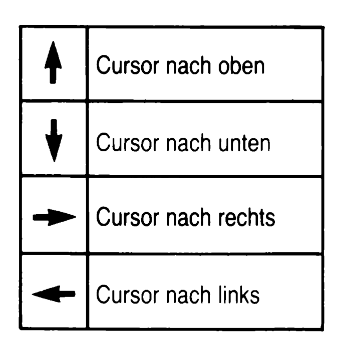
Es empfiehlt sich, die Zeilen mit 10er Schritten zu belegen, d.h. die Zeilen mit 10, 20, 30 usw. durchzunummerieren, denn bei eventuellem Einfügen verschiedener Modifikationen im Programm stehen zwischen den 10er Schritten die 1er zur Verfügung, um Ergänzungen in das Programm aufzunehmen.

2.4 Cursor und Cursorsteuerung

Nach Einschalten Ihres Commodore-Gerätes erscheint (nach der READY-Meldung) auf dem Monitor ein schwarzer Punkt, der blinkt. Dieser blinkende Punkt wird als **„Cursor"** bezeichnet. Er gibt an, an welcher Stelle Sie Symbole auf dem Monitor eintippen können, die Sie dann mit der Taste RETURN in die Zentraleinheit weitergeben.

Um die richtige Wirkung zu erzielen und um den Cursor an der richtigen Stelle zu haben, muß es spezielle Cursor-Steuerungstasten geben.

Dies sind die Tasten (rechts auf Ihrem Gerät), auf denen sich Pfeile befinden, die senkrecht nach oben und nach unten und vertikal nach links und nach rechts zeigen. Sie bewegen den Cursor an diejenige Stelle, an der wir schreiben wollen. Im einzelnen bewirken diese Pfeile folgendes:



↑	Cursor nach oben
↓	Cursor nach unten
→	Cursor nach rechts
←	Cursor nach links

Abb. 9: Cursor und Cursorsteuerung

Beachten Sie bitte, daß bei den Commodore-Geräten eine sog. REPEAT-Funktion vorhanden ist, d.h. das Zeichen oder die Funktion wird, solange die Taste gedrückt bleibt, wiederholt. Damit läuft der Cursor über den Bildschirm nach oben, nach unten, nach rechts oder links, so lange, solange die Taste gedrückt ist; man erspart sich damit mühevolles Wiederholen des Tastendrucks.

Weitere Symbole der Cursor-Steuerung sind die Tasten, die sich rechts oben am Gerät befinden und beschriftet sind mit HOME/CLEAR und DEL/INST.

Das einfache Drücken der Taste HOME/CLEAR bewirkt, daß der Cursor in die obere linke Ecke springt, die sog. HOME-Position. Der Monitorinhalt bleibt erhalten.

Wird die Taste HOME/CLEAR gemeinsam mit der Taste SHIFT gedrückt, so tritt die CLEAR-Funktion ein; der Cursor springt in die linke obere Ecke des Monitors, wobei zugleich der gesamte Bildschirminhalt gelöscht wird (die gleiche Wirkung wird durch eine „Kurzeingabe" erzielt).

Oftmals ist notwendig, Buchstaben, Ziffern und/oder Zeichen in einem bereits vorgegebenen Text einzufügen, weil man sich entweder vertippt hat oder diese Zeichen vergessen hat. Dies kann erreicht werden durch die Taste DEL/INST. Wenn Sie nur diese Taste drükken, so wird dasjenige Zeichen, das links vom Cursor steht, gelöscht. Der Cursor nimmt diejenige Position ein, auf der dieses Zeichen, das gelöscht wurde, ehemals stand.

Wenn Sie diese Taste gemeinsam mit der SHIFT-Taste drücken, so bleibt der Cursor an der vorgegebenen Position stehen, das Symbol, auf dem er stand, bewegt sich einschließlich des gesamten Zeileninhalts nach rechts, somit entsteht Platz für das Einfügen von Buchstaben, Ziffern und/oder Zeichen oder Worten oder Sätzen.

Im einzelnen bewirken die Tasten folgendes:

TASTEN	WIRKUNG
HOME/CLEAR	Der Cursor springt in die linke obere Ecke des Monitors
SHIFT + HOME/CLEAR	Der Cursor springt in die linke obere Ecke und löscht den Bildschirminhalt
SHIFT + F7	dto. (Kurzeingabe)
DEL/INST	Der Cursor springt auf das links neben ihm stehende Zeichen und löscht es
SHIFT + DEL/INST	Der Cursor schiebt das Zeichen, auf dem er gerade steht, nach rechts, wobei sich die gesamte Zeile ab Cursorposition verschiebt.

Abb. 10: Weitere Cursorsteuerung

Die Cursor-Steuertasten präge man sich sorgfältig ein, da sie zur sofortigen Korrektur von Falscheingaben bzw. von offensichtlichen Fehlern und Tippfehlern unbedingt erforderlich sind.

Es ist aber auch möglich, auf dem Monitor das Schriftbild umzukehren, d.h. helle Symbole auf dunklem Hintergrund zu drucken. Dies geschieht durch die Tasten INV.ON und INV.OFF (wobei: INV. = invers) und dem gleichzeitigen Drücken der Taste CONTROL.

2.5 Weitere Systemkommandos

Sind in einem BASIC-Programm Fehler, so gibt der Computer jeweils eine entsprechende **Fehlermeldung** aus. Einer der wesentlichen Fehler am Anfang wird immer ein sog. „SYNTAX ERROR" sein. Dies bedeutet, daß die Schreibweise, ein Symbol oder ein Zeichen, falsch ist oder am falschen Platz steht. Die angegebene Zeilennummer ist zu prüfen und der Fehler zu korrigieren.

Die Fehlermeldungen und ihre Bedeutung im einzelnen entnehmen Sie bitte dem Anhang.

Wenn der Bildschirm durch SHIFT+CLEAR/HOME freigemacht wurde und sich beim Ablaufen des Programms ein Fehler ergibt, so muß es möglich sein, die fehlerhafte Zeile oder den fehlerhaften Teil des Programm auf den Bildschirm zu rufen.

Dies geschieht durch das Systemkommando: LIST [RETURN]

Im einzelnen sind folgende Eingaben, die jeweils mit der RETURN-Taste abzuschließen sind, möglich:

LIST Zeigt das gesamte Programm

LIST 100- Zeigt das Programm von Zeile 100 bis zum Programmende

LIST 10 Zeigt nur die Programmzeile 10

LIST -100 Zeigt das Programm vom Programmanfang bis einschließlich Zeile 100

LIST 10-200 Zeigt alle Programmzeilen ab Zeile 10 bis inclusive Programmzeile 200

HINWEIS: Da der Bildschirm nur eine begrenzte Anzahl von Zeilen zuläßt, lassen sich meist nur Teile des Programms darstellen.

Wurde ein Programm geschrieben und wurde es vollständig in den Commodore eingegeben, so ist ein besonderes Systemkommando erforderlich, um es abarbeiten zu lassen. **Ein Programm wird durch den Befehl RUN und dem anschließenden Drücken der RETURN-Taste gestartet.**

RUN [RETURN]

Jetzt wird Zeile für Zeile, beginnend bei der niedrigsten Zeilennummer, das Programm abgearbeitet.

Hat man ein Programm abgearbeitet und möchte ein anderes Programm eingeben, so ist der Arbeitsspeicher „frei" zu machen, da immer nur e i n Programm abgearbeitet werden kann. Dieses „frei-machen" geschieht durch das Systemkommando NEW, das durch die RETURN-Taste abgeschickt wird.

RETURN

2.6 Kontrollfragen

1. Was bedeutet die Monitormeldung nach dem Einschalten:
   COMMODORE BASIC V3.5 ...?

2. Was ist ein BYTE?

3. Erklären Sie sich die Begriffe:
   - Kommando und Anweisung
   - Funktionen
   - Operatoren
   - Satz- und Sonderzeichen

4. Welches ist der wesentliche Unterschied zwischen einem Systemkommando und einem BASIC-Befehl?

5. Welche Bedeutung hat der „CURSOR"?

6. Machen Sie sich mit den CURSOR-Steuertasten vertraut!

7. Wie wird die ZE des Computers für ein neues Programm freigemacht?

8. Wie wird ein Programm gestartet?

9. Wie kann man das Programm auf dem Monitor sichtbar machen?

Für Ihre Lösungen:

3. Grundlegende Ein- und Ausgabeanweisungen

Lernziel: Der Leser soll die grundlegenden Eingabe- und Ausgabeanweisungen kennen und benutzen lernen und die dafür notwendigen Konstanten und Variablen in ihrer Bedeutung verstehen.

3.1 Variable und Konstante

Der BASIC-Befehl besteht aus den Bestandteilen Anweisungsnummer, BASIC-Schlüsselwort und Operandenfeld. Im Operandenfeld eines jeden Befehls können Konstante und/oder Variable aufgeführt sein.

Konstante sind dabei diejenigen Daten eines BASIC-Programms, die sich nicht verändern. Grundsätzlich wird unterschieden zwischen

1. Numerischen Konstanten (= Zahlenwerte)

   und

2. Zeichenkettenkonstante "Strings" (= Worte oder Zahlenwortkombinationen).

Numerische Konstante:		
1. Ganzzahl-Konstante (INTEGER)	Kennzeichen	= %
	Wertebereich	= -32767 bis 32767
	Speicherplatz	= 7 Bytes
	Beispiele:	543, -12000, -1
2. Dezimal-Konstante (REAL)	Kennzeichen	= keines
	Wertebereich	= größer als INTEGER z.B.: 999999
		= oder Punkt mit 7 Stellen z.B.: 0.00005, .543, 567.891
		= oder Exponent z.B.: 4.2E6 = $4.2*10^6$
	Speicherplatz	= 7 Bytes
Zeichenkette-Konstante (STRINGS)	Kennzeichen	= $
	Wertebereich	= bis max. 255 Zeichen
	Speicherplatz	= bis max. 255 Bytes
	Beispiele:	„Commodore C116" „DM" „Name Vorname Wohnort"

Abb. 11: Konstante

Variable sind symbolische Namen und damit Platzhalter für einen Wert, der diesen Variablen zugewiesen werden soll. Vergleichbar sind die Variablen mit jeder Rechenformel, die unter Verwendung von Buchstabensymbolen aufgestellt wird und denen bestimmte Zahlenwerte zugewiesen werden, wenn die Formel berechnet werden soll. In der Computertechnik nennt man diese Symbole Variable.

Jeder im Programm angesprochenen Variablen wird in der Zentraleinheit (Arbeitsspeicher) ein ganz besonderer Speicherplatz zugewiesen, in dem während des Programmlaufs beliebige Werte abgespeichert werden können. Wird später im Programm die betreffende Variable aufgerufen, so holt der Computer aus der dazugehörigen Speicherstelle den dort abgelegten Wert und arbeitet mit diesem so, wie es der Programmbefehl vorsieht.

Als Symbole für die Namen der Variablen dienen die Buchstaben von A - Z, Buchstabenkombinationen und Buchstabenzahlenkombinationen.

1)	Ein Buchstabe:	A bis Z	26 Möglichkeiten
2)	Zwei Buchstaben:	AA, AB, ... ZZ	676 Möglichkeiten
3)	Ein Buchstabe und eine Ziffer:	A0, A1, ... Z9	260 Möglichkeiten
Insgesamt mögliche Variablennamen:			962

HINWEIS: Beachten Sie bitte das Handbuch!
Einige Kombinationen sind nicht möglich!
z.B.:

FN, IF, ON, OR, TO	BASIC-Befehlsworte
TI	laufende Zeit
ST	Status.

Abb. 12: Variablen-Namen

Bei den Variablen unterscheiden wir ebenso wie bei den Konstanten zwischen Ganzzahlvariablen, Dezimalvariablen und Zeichenketten- bzw. Textvariablen.

Numerische Variable:				
1. Ganzzahl-Variable	Kennzeichen		=	%
(INTEGER = INT)	Beispiele:	A%	=	17
		A1%	=	123
		AB%	=	0
2. Dezimal-Variable	Kennzeichen		=	keines
(REAL)	Beispiele:	A	=	17,456
		A1	=	123,45
		AB	=	0,76
3. Zeichenkette-Variable	Kennzeichen		=	S
(STRING)	Beispiele:	A$	=	„Commodore C116"
		A1$	=	„DM"
		AB$	=	„Name Vorname"

Abb. 13: Variable

Da die Variablen Platzhalter sind, müssen ihnen Werte zugewiesen werden. **Einer Ganzzahl- bzw. Realzahlvariablen wird, wenn kein Wert vom Programmierer zugewiesen wird, vom Computer der Wert Null eingesetzt.** Neben dem Zuweisen von Werten werden in Variablen aber auch die Ergebnisse von Rechnungen bzw. von Formeln festgehalten.

In String-Variablen werden Worte oder ganze Sätze festgehalten.

Natürlich können unter den zugewiesenen Namen die abgespeicherten Werte wieder aufgerufen und verarbeitet werden.

3.2 Eingabe mit INPUT

BASIC-Schlüsselwort: INPUT

Struktursymbol:

Anweisung

Beispiel: 10 INPUT A

```
10 INPUT A
|    |   |
|____|___|________ Anweisungsnummer
     |   |
     |___|________ Schlüsselwort
         |
         |________ Operandenfeld
                   (= Variablenname)
```

Monitor: Auf dem Monitor erscheint ? und eine Antwort hat zu erfolgen:

? 500 RETURN

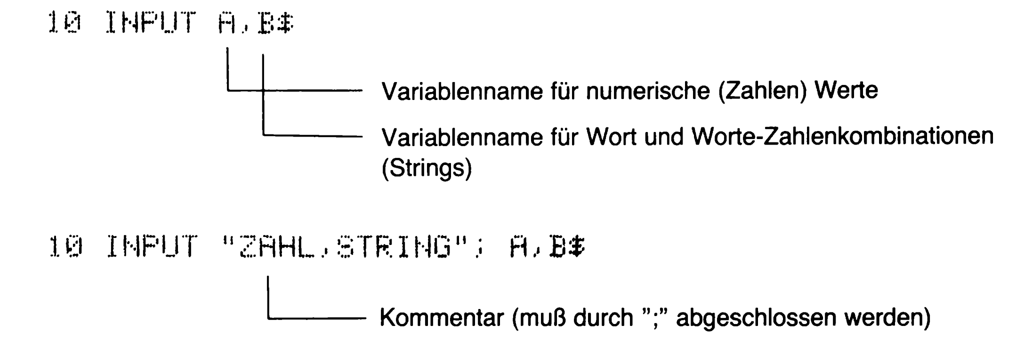
Die Variable A hat damit den Inhalt 500.

Der zweite Teil eines jeden BASIC-Befehls ist das Schlüsselwort. **Das Schlüsselwort INPUT bietet die Möglichkeit, in einem Programm eine oder mehrere variable Werte an die Zentraleinheit des C116/16/+4 zu übergeben.** Wenn eine BASIC-Zeile erreicht wird, in der das Wort INPUT steht, so hält das Programm an und gibt auf dem Monitor ein Fragezeichen (?) aus und wartet, bis ein Wert bzw. ein String eingegeben wird. Wenn die Eingabe auf dem Monitor erfolgt ist, so ist sie mit der Taste RETURN abzuschließen. Danach läuft das Programm mit der nächsten Zeile weiter.

Dem BASIC-Befehlswort INPUT muß der Name der Variablen folgen; zwischen INPUT und Variable-Name steht kein Zeichen. Es können aber auch mehrere Variable folgen, die dann durch Kommata getrennt werden müssen.

Da das Wort INPUT lediglich die Ausgabe eines Fragezeichens auf dem Monitor bewirkt, ist es oftmals für den Benutzer wissenswert, welcher Wert eingegeben werden soll. Eine Unterstützung hierzu bietet die Möglichkeit, nach dem Schlüsselwort INPUT einen Kommentar einzufügen. Dieser Kommentar steht in Anführungsstrichen (") und muß mit einem Semikolon (;) von dem Namen der Variablen getrennt werden.

Sollen mehrere Variable hintereinander eingegeben werden, so erwartet das Gerät genau so viel Werte wie Variable vorhanden sind. Die Eingabe der Werte hat dann getrennt durch Komma zu erfolgen.



HINWEIS: Beachten Sie, daß in der amerikanischen Schreibweise das Komma ein Punkt ist.

Werden weniger Werte eingegeben als Variable vorhanden sind, so werden die fehlenden Werte durch zwei Fragezeichen nachgefordert. Werden mehr eingegeben als gefordert sind, so erscheint die Fehlermeldung EXTRA IGNORED und das Programm läuft ohne die zuviel eingegebenen Werte weiter.

3.3 Ausgabe mit PRINT

Schlüsselwort: PRINT

PRINT-USING

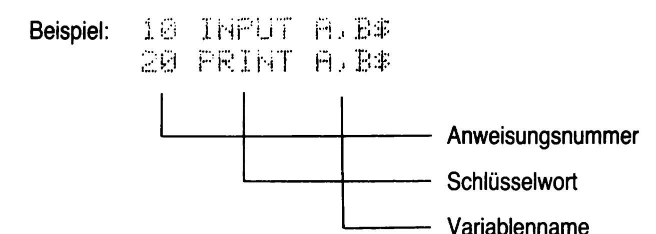
TAB

SPC

Struktursymbol:

Anweisung

Beispiel:



Monitor: Die Variablen A und B sind nach dem ? mit Werten zu füllen, die durch die Zeile 20 auf Monitor nebeneinander ausgegeben werden.

3.3.1 Das Schlüsselwort PRINT

Die PRINT-Anweisung ist die wichtigste Ausgabenanweisung im BASIC. Sie bietet eine Vielzahl von Möglichkeiten der Bildschirm- bzw. der Druckgestaltung.

Nach dem Wort PRINT kann stehen:

1. Auszugebende Texte innerhalb von Anführungszeichen

Die Texte innerhalb der Anführungszeichen werden wortwörtlich wiedergegeben und damit genauso ausgedruckt, wie sie hinter der PRINT-Anweisung stehen.

2. Namen von Variablen

Die Variablen-Namen (z.B. A, A1, AB) werden mit ihrem Wert ausgedruckt. Dabei ist es gleichgültig, ob es sich um Zahlenwerte oder Strings handelt.

3. Funktionen

Das Ergebnis der Formeln ist in einer Variablen gespeichert; der Inhalt wird ausgedruckt.

4. Satzzeichen

Das Komma: Das Komma teilt den Bildschirm für den Ausdruck von Daten in vier Bereiche ein (im einzelnen siehe Handbuch).

Das Semikolon: Bei Verwendung des Semikolons werden die Daten ohne Zwischenraum gedruckt. Dies gilt für Worte. Werden jedoch Variable ausgedruckt, die Zahlenwerte beinhalten, so beträgt der Zwischenraum eine Freistelle und eine zweite Stelle für das Vorzeichen (das Plus wird nicht ausgedruckt).

Beispiel:

```
10 INPUT A,B$
20 PRINT A,B$
30 PRINT "ZAHL"A"WORT"B$
40 PRINT "ZAHL";A;"WORT";B$
50 PRINT "ZAHL",A,"WORT",B$
```

Kommentar

"," und ";" führen zu unterschiedlichen Ausgaben

HINWEIS: Jede Zeile mit RETURN absenden !!!

Das Gesamtprogramm ist mit RUN RETURN zu starten.

3.3.2 Besonderheiten zum BASIC-Schlüsselwort „PRINT"

Es ist möglich, mit dem Commodore-Gerät **direkt Rechnungen** auszuführen. Dazu ist es erforderlich, das BASIC-Schlüsselwort PRINT als Systemkommando zu verwenden. Soll beispielsweise die Addition 3 + 4 ausgeführt werden, so ist folgende Befehlsvorgabe notwendig:

PRINT3+4 RETURN

Das Ergebnis 7 wird auf dem Bildschirm ausgegeben, und unter dem Ergebnis steht das Wort READY, darunter wieder der blinkende Cursor. Die Zahlen werden dabei auf sieben Dezimalstellen hinter dem Komma genau wiedergegeben. Bitte beachten Sie, daß statt des Kommas der Punkt steht. Die Rechenanweisungen müssen in PRINT-STATEMENTS ohne Anführungszeichen geschrieben werden.

Dabei gilt folgende Rangfolge für das Ausführen der Rechenoperationen:

Rang 1:	(	)	Klammer
Rang 2:	–		Negative Zahl (Vorzeichen)
Rang 3:	↑		Potenzieren
Rang 4:	*	/	Multiplizieren/Dividieren
Rang 5:	+	–	Addieren/Subtrahieren

Abb. 14: Rangfolge bei der Ausführung von Rechenoperationen

Es ist möglich, anstelle des BASIC-Schlüsselworts „PRINT" auch ein Fragezeichen "?" zu schreiben. Dieses Fragezeichen ersetzt das Schlüsselwort PRINT und hat die gleiche Wirkung.

HINWEIS: Es darf bei Ausgaben auf dem Drucker nicht benutzt werden!

Beispiel 1:

```
READY
?3+7          [RETURN]
7
READY
C
```

Beispiel 2:

```
READY
?22+10/2      [RETURN]
9
READY
C
```

Beispiel 3:

```
?(22+10)/2    [RETURN]
7
READY
C
```

Will man in einem Programm eine **Leerzeile** erzeugen, so geschieht das dadurch, daß man das Operandenfeld hinter dem Schlüsselwort PRINT einfach wegläßt. Die Anweisung lautet:

10 PRINT

Der Computer geht davon aus, daß das, was hinter dem BASIC-Schlüsselwort „PRINT" steht, entweder auf dem Monitor oder dem Drucker gedruckt werden soll. Da hinter dem Schlüsselwort keine Anweisung enthalten ist, ist im Betriebssystem vorgesehen, daß eine Zeile lang „nichts" zu drucken ist; es entsteht eine Leerzeile.

Die Anweisungsnummer als Bestandteil eines BASIC-Befehls kann fehlen, wenn ein Befehl direkt hinter einem anderen Befehl angeschlossen wird und dieser Anschluß durch Doppelpunkte erfolgt; z.B.:

10 PRINT:PRINT

Diese Anweisung bewirkt, daß der Computer unter der Anweisungsnummer 10 zweimal den PRINT abarbeitet. Beide Male ist hinter dem PRINT-Befehl kein Operandenfeld, so daß der Computer zwei Leerzeilen druckt.

Für einen Anfänger ist es nicht empfehlenswert, kompliziertere Zusammenhänge unter einer Zeilennummer, getrennt durch Doppelpunkte, zu programmieren. Die Doppelpunkte erschweren nämlich die Fehlersuche.

Die SPACE-(SPC)Funktion wird in Verbindung mit der PRINT-Anweisung verwendet, um eine bestimmte Anzahl (X) von Stellen zu überspringen. X kann dabei Werte zwischen 0 und 255 annehmen.

Beispiel:
```
10 PRINT "COMMODORE" SPC(6) "116"
```

Die TAB-Funktion (TAB) wird ebenfalls in Verbindung mit der PRINT-Anweisung benutzt. Das Druckzeichen, das der TAB-Anweisung folgt, wird in derjenigen Spalte gedruckt, die dem Wert X entspricht. D.h. es wird bei jeder TAB-Anweisung vom Zeilenanfang gezählt.

Beispiel:
```
10 PRINT TAB(16)"COMMODORE 116"
```

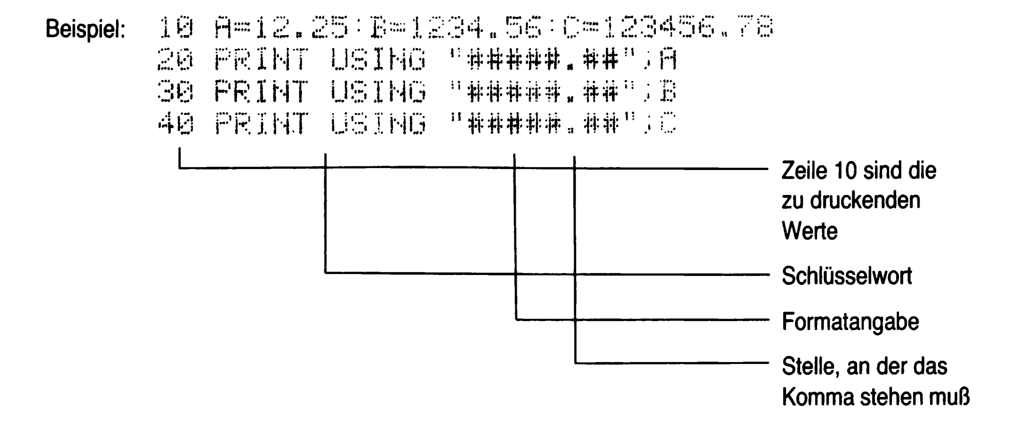
3.3.3 Formatierte Ausgabe mit PRINT USING

Die PRINT-Anweisung bewirkt, daß die ausgegebenen Zeichen vom Zeilenanfang geschrieben werden. Dabei wird als Schreibweise immer linksbündig geschrieben, d.h. es stehen bei DM-Beträgen beispielsweise die Kommata nicht untereinander. Um nun diese komma-orientierte Schreibweise an verschiedenen Positionen des Monitors oder des Druckers zu ermöglichen, können Formate vorgegeben werden. **Die PRINT USING-Anweisung ist also ein Anweisung, die ein gewünschtes Format vorgeben kann.**

Die Formatangabe steht grundsätzlich in Anführungsstrichen hinter der PRINT USING-Anweisung. Das Format selbst wird in Anführungsstriche gesetzt und durch verschiedene Symbole als Platzreservierung dargestellt.

HINWEIS: Zur Benutzung unterschiedlicher Symbole und ihrer Wirkungsweise schauen Sie bitte im einzelnen im Bedienungshandbuch nach.

Beispiel:



Monitor:

```
   12.25
 1234.56
**********
```

Der Inhalt von C ist für das Format zu lang, deshalb wird das Feld mit "*" ausgefüllt.

3.4 BEISPIEL: Das BASIC-Programm „einfache Zinsberechnungen"

Das Beispiel der einfachen Zinsrechnung soll zu jedem beschriebenen Kapitel erneut als Demonstration benutzt werden. Aus diesem Grund soll die folgende Problemanalyse nachvollzogen werden, da in den späteren Kapiteln darauf Bezug genommen wird.

1. Problemstellung:
   Das folgende Programm soll die Möglichkeit bieten, einfache Bankzinsen per Formel zu berechnen.

2. Problemanalyse:
   Im Gegensatz zur Zinseszinsformel bedeuten einfache Bankzinsen, daß die Zinsen, wie bei einem Sparbuch, Jahr für Jahr berechnet werden und entsprechend ihrer Laufzeit, auch unterteilt nach Tage, gutgeschrieben werden.

3. Der Lösungsweg:
   Dies ist hier ein einfaches Problem, da wir lediglich die Bankzinsformel anwenden müssen; sie lautet:

   Z = K0 x P x T/(1000 x 360)

   daraus folgt, daß die Werte für die Variablen einzugeben sind und der Computer dann die Formel berechnet. Einzugeben sind: Kapital, Zinssatz und Laufzeit.

   Die Berechnung soll so erfolgen, daß dem Benutzer des Programms das Neue Kapital und die gewährten Zinsen mitgeteilt werden, d.h. die Zinsen sind nach der Berechnung zu dem Kapital hinzuzufügen.

Variable:			
	K0	=	Kapital
	P	=	Zinssatz
	T	=	Laufzeit in Tagen
	KN	=	Neues Kapital
	Z	=	Zinsen

4. Der Lösungsalgorithmus (= Struktogramm):
   Das Programm soll so ablaufen, daß wir die Werte als Variable für die Formel eingeben, die Formel berechnen lassen und die entsprechend berechneten Zinsen ausgeben. Als zusätzlichen Service bieten wir die Berechnung des neuen Kapitals, d.h. die Addition der Zinsen zum Anfangskapital.

5. Die Codierung:
   An und für sich, bei einem größeren Programm unbedingt, sollten wir uns jetzt an den Schreibtisch setzen und das Programm in BASIC auf ein Stück Papier notieren. Bei dialog-orientierten Systemen, d.h. über den Monitor, wird angezeigt, welche Schritte wir unternehmen, deshalb kann man sich bei kleineren Programmen diesen Schritt ersparen und sofort mit Punkt 6 beginnen.

6. Die Implementierung:
   Das codierte Programm wird in die Maschine eingegeben.

   HINWEIS: Im folgenden werden wir uns speziell mit diesem Punkt 6 Implementierung beschäftigen.

7. Testen des Programms und Fehler beseitigen:
   Auch hiermit werden wir uns beschäftigen, doch sei bereits jetzt auf Kapitel 9 „Fehlersuche und Fehlerbeseitigung" verwiesen.

8. Das lauffähige Programm steht zum Einsatz bereit.

Man bezeichnet die Phasen Codierung – Implementierung – Test mit Fehlerbeseitigung als die sog. „Realisierungsphase" in der Programmierung. Mit dieser Phase wollen wir uns im folgenden speziell beschäftigen.

LÖSUNG: STRUKTOGRAMM:

SCHREIBEN ÜBERSCHRIFT
DRUCKE LEERZEILE
EINGABE ANFANGSKAPITAL
EINGABE ZINSSATZ
EINGABE LAUFZEIT
BERECHNEN ZINSEN
BERECHNEN NEUES KAPITAL
DRUCKEN LEERZEILEN
AUSGABE NEUES KAPITAL
AUSGABE ZINSEN
ENDE

```
10 PRINT "EINFACHE ZINSRECHNUNG"
15 PRINT
20 INPUT "ANFANGSKAPITAL   ";K0
30 INPUT "ZINSSATZ         ";P
40 INPUT "LAUFZEIT IN TAGEN";T
50 Z=K0*P*T/(360*100)
60 KN=K0+Z
65 PRINT:PRINT
70 PRINT USING "############.##";"NEUES KAPITAL =",KN
80 PRINT USING "############.##";"ZINSEN =";Z
90 END
```

BEISPIELHAFTER PROGRAMMABLAUF:

Programmablauf/Monitor:

RUN [RETURN]

```
EINFACHE ZINSRECHNUNG

ANFANGSKAPITAL      ? 1000
ZINSSATZ            ? 10
LAUFZEIT IN TAGEN   ? 360

NEUES KAPITAL       = 1100
Zinsen              =  100

READY
C
```

3.5 Kontrollfragen

1. Welches ist der Unterschied zwischen einer Variablen und einer Konstanten

2. Welche Namen können Variable und Konstante haben?

3. Machen Sie sich die Begriffe INTEGER-, Dezimalzahl- und STRING-Variable klar! Welche Sonderzeichen müssen Sie im Programm haben?

4. Schreiben Sie ein kleines Programm, bei dem 2 Zahlen eingegeben und wieder ausgegeben werden.

5. Welche Rangfolge gilt bei Rechenoperationen?

6. Schreiben Sie ein kleines Programm, das 2 Zahlen addiert und das Ergebnis ausgibt!

7. Geben Sie das Programm „Einfache Zinsrechnung" ein und führen Sie den Programmlauf aus.

8. Welche Bedeutung hat die [RETURN] -Taste?

Für Ihre Lösungen:

4. Grundlegende Strukturanweisungen

Lernziel: Der Leser soll die grundlegenden BASIC-Schlüsselworte, die ein Programm strukturieren, kennenlernen und im Programm anwenden können.

BASIC-Schlüsselwort: GOTO

FOR ... TO ...
NEXT

DO-WHILE LOOP etc.

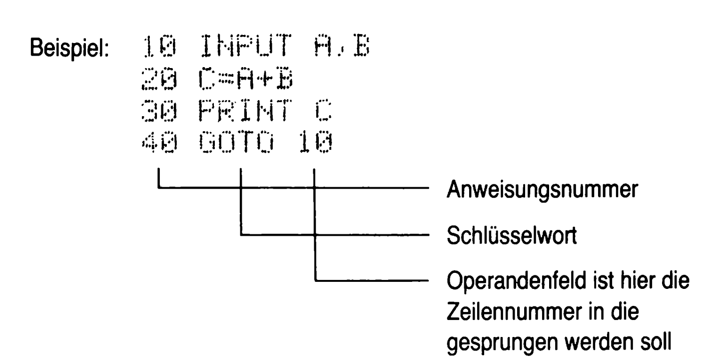
4.1 Der unbedingte Sprung mit GOTO

BASIC-Schlüsselwort: GOTO

Struktursymbol:

Anweisung

Sprung
(hier: Rücksprung)



Das BASIC-Schlüsselwort GOTO bewirkt, daß zu derjenigen Zeile gesprungen wird, die hinter dem Schlüsselwort GOTO steht. Diese GOTO-Anweisung bewirkt einen ständigen Programmablauf (unendliche Schleife), und zwar so lange, bis die Taste RUN/STOP gedrückt wird.

Da die GOTO-Anweisung bedingungslos verzweigt, macht das Zwischenspringen mit GOTO ein Programm nicht übersichtlicher, sondern im Gegenteil, es entstehen „wilde Sprünge", die ein Programm schwer lesbar und verstehbar machen.

HINWEIS: Das Schlüsselwort GOTO ist einfach handzuhaben, verführt aber zu einem nicht strukturierten, unübersichtlichen Programmieren, deshalb sollte die Anwendung dieses Schlüsselwortes auf ein Minimum beschränkt werden.

Bei dem Commodore C116/C16/+4 kann das Schlüsselwort GOTO auch als Systemkommando, d.h. ohne Anweisungsnummer benutzt werden und bewirkt einen Programmstart ab der eingegebenen Zeilenzahl. Dies ist besonders bei dem Testen von Programmen interessant, da dadurch ab der angegebenen Zeilennummer das Programm gestartet werden kann, ohne daß die Inhalte vorher aufgefüllter Variabler zerstört werden (dies ist bei dem Wort RUN der Fall).

4.2 Programmschleife

BASIC-Schlüsselwort: FOR .... TO ....

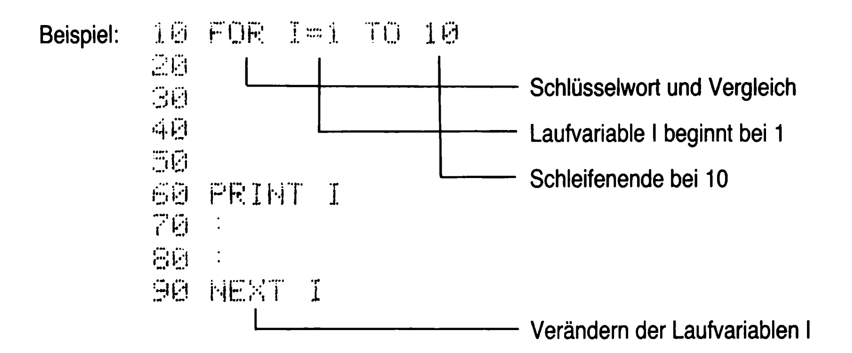
NEXT

Struktursymbol:

Wiederhole bis Schleifende

Anweisung

Beispiel:



Oftmals ist es erforderlich, daß der Computer ein bestimmtes Programm einer bestimmten Anzahl entsprechend abarbeitet. Dabei muß eine Variable zählen, wie oft die Schleife durchlaufen wird. Es ist die sogenannte Zählvariable oder auch Laufvariable.

Dabei gilt die folgende Logik:

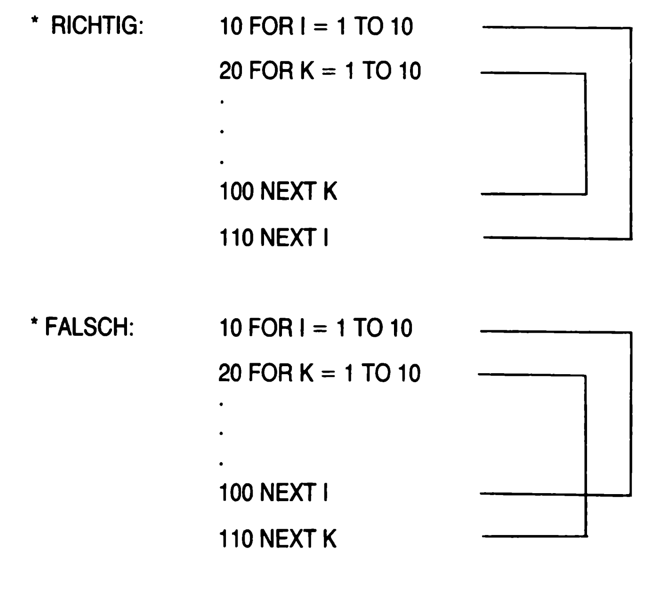
Zuerst wird die Laufvariable auf den Startwert gesetzt. Wird die NEXT-Anweisung erreicht, so wird die Laufvariable um eins erhöht und danach wird das Ergebnis mit dem Endwert verglichen. Ist das Ergebnis kleiner oder gleich, wird die unmittelbar der FOR-Anweisung folgende Programmzeile ausgeführt.

HINWEIS: Ein Vergleich auf größer wird nicht durchgeführt und kann bei falscher Berechnung zu Fehlern führen.

Die Schrittweise der FOR-NEXT-Schleife ist standardgemäß eins; durch Ergänzung um das Wort STEP kann die Schrittweise verändert, d.h. entweder kleiner oder auch größer gemacht werden.

Verschachtelte Schleifen:

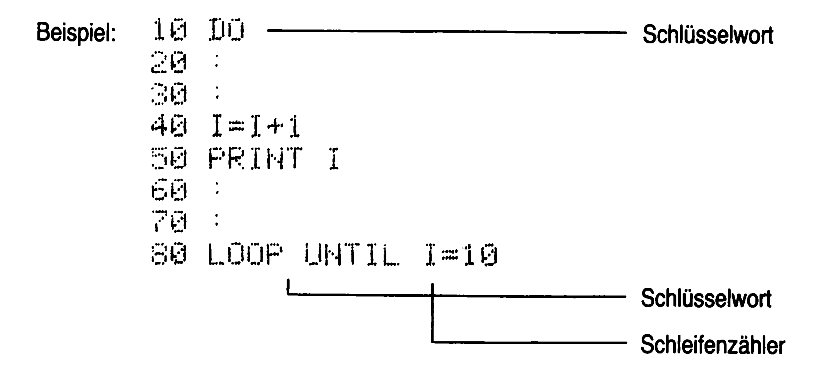
Verschachtelte Schleifen müssen so angelegt werden, daß die äußere Schleife erst dann weiter zählt, wenn die innere ganz abgearbeitet ist.



4.3 Besondere Schleifensteuerung mit DO-WHILE-LOOP etc.

BASIC-Schlüsselwort: DO

LOOP UNTIL



Schlüsselwort: DO WHILE

LOOP

Beispiel:

```
10 DO WHILE I<10
20 :
30 :
40 I=I+1
50 PRINT I
60 :
70 :
80 LOOP
```

DO WHILE — Schlüsselwort

I<10 — Schleifenzähler

LOOP — Schlüsselwort

Die o.g. BASIC-Schlüsselworte sind Besonderheiten des Commodore-Systems, die die Schleifensteuerung erheblich erleichtern.

Der DO-WHILE-LOOP dient dazu, um eine versuchte Wiederholung des Abarbeitens der Schleife abzuweisen. Man spricht daher auch von einer abweisenden Schleife, da sie die Wiederholung des Wiederholungsteils abweisen kann. Es ist ein Schleifentyp „solange-tue-Schleife".

Das Befehlswort DO WHILE sollte damit am Anfang der Schleife stehen, um die Bedingung, den Durchlauf der Schleife abzuweisen, erfüllen zu können.

Die Schleife mit dem BASIC-Befehlswort DO-LOOP UNTIL bezeichnet man als nicht abweisende Schleife; sie sollte am Ende desjenigen Teiles im Programm stehen, der immer wiederholt werden soll. **Diese „wiederhole-bis-Schleife" bricht die Wiederholung am Ende der Schleife ab.** Damit sollte das Wort UNTIL auch am Ende der Schleife stehen. Allerdings ist dies nicht unbedingt notwendig, dort wird aber die Bedingung der nicht abweisenden Schleife exakter erfüllt.

Manchmal ist es notwendig, die Schleifenabfrage nicht am Beginn oder am Ende der Schleife unterzubringen, sondern irgendwo in der Mitte des Wiederholungsteils. Dieser Schleifentyp kann für den Commodore C116/C16/+4 mit der BASIC-Anweisung DO-LOOP-EXIT programmiert werden. **Der Wiederholungsteil wird dann verlassen, wenn die gesetzte Bedingung erfüllt ist;** das Wort EXIT bewirkt, daß das Programm in derjenigen Zeile fortgesetzt wird, die hinter der LOOP-Zeile steht.

4.4 BEISPIEL: „Einfache Zinsrechnung" mit Rücksprung

Wir können davon ausgehen, daß eine große Anzahl Kunden nicht nur mit einem einzigen Kapitel anfragen, sondern alternative Berechnungen wünschen.

Problemanalyse:

Wir müssen das Programm mehrfach abarbeiten und zwar je nachdem, wieviele Kapitalien zu berechnen sind.
Gehen wir davon aus, daß die Kunden die genaue Anzahl der Kapitalien wissen, so schließt sich eine GOTO Verzweigung aus.
Wir lösen die Aufgabe mit einer FOR ... TO ... -Schleife.

Die Ausgabe soll auf einem freien Monitor erfolgen, deshalb programmieren wir in Zeile 1 die Funktionstaste „SHIFT + CLEAR/HOME".

Variable:

Zusätzlich zu den bereits bekannten Variablen benötigen wir

I = Schleifenzähler
N = Anzahl der Kapitalien.

STRUKTOGRAMM:

MONITOR FREI / ÜBERSCHRIFT
EINGABE ANZAHL KAPITALIEN
WIEDERHOLE BIS N
EINGABE KØ, P, T
BERECHNEN Z, KN
AUSGABE KN, Z
ENDE

CODIERUNG:

```
1 PRINT "\u2758"
10 PRINT "EINFACHE ZINSRECHNUNG"
20 PRINT
30 INPUT "WIEVIELE KAPITALIEN?";N
40 PRINT
50 FOR I=1 TO N
60 INPUT "ANFANGSKAPITAL?";K0
70 INPUT "ZINSSATZ       ?";P
80 INPUT "LAUFZEIT/TAGE ?";T
90 Z=K0*P*T/(360*100)
100 KN=K0+Z
110 PRINT
120 PRINT "NEUES KAPITAL =",KN,"DM"
130 PRINT "ZINSEN        =",Z,"DM"
140 PRINT
150 NEXT I


READY.
```

4.5 Kontrollfragen

1. Weshalb sollte das BASIC-Schlüsselwort GOTO vorsichtig benutzt werden?

2. Was ist eine Programmschleife?

3. Wie dürfen Programmschleifen nur aufgebaut sein?

4. Welche besonderen Formen der Programmschleifen bietet der Commodore 116/16/+4?

5. Schreiben Sie ein Programm, das die Zahlen von 1 bis 10 addiert!

6. Welche Bedeutung hat der Schleifenzähler?

7. Verändern Sie Programm zu Frage Nr. 5 durch Einfügen einer Variablen so, daß beliebig viele Zahlen addiert werden können!

8. Welche Taste steuert den CURSOR innerhalb von "...." Anführungsstrichen?

Für Ihre Lösungen:

5. Wertezuweisung und logische Operationen

Lernziel: Der Leser soll weitere Verzweigungstechniken kennen- und anwenden lernen.

BASIC-Schlüsselwort: LET

AND

OR

NOT

IF ... THEN ...

IF ... THEN ... ELSE ...

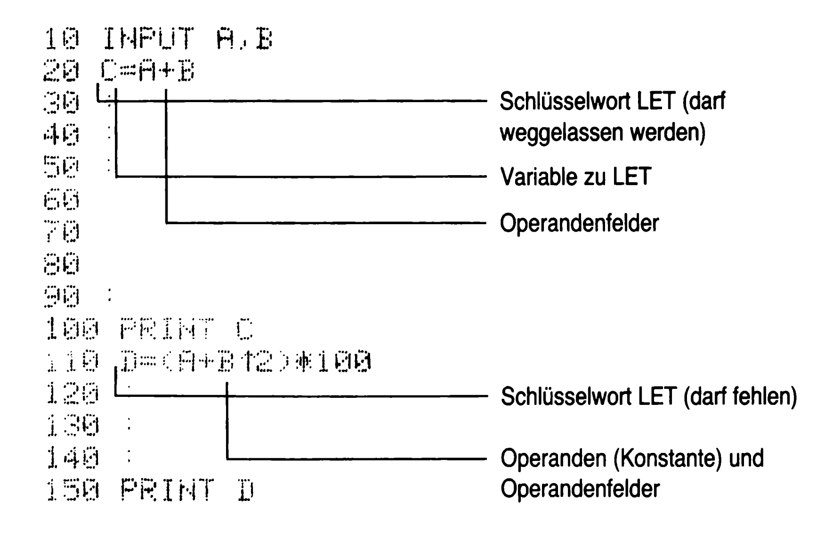
5.1 Arithmetische Operatoren

BASIC-Schlüsselwort: LET

Struktursymbol:

Anweisung

Beispiel: Rechnen mit zwei über INPUT eingegebenen Zahlen



Start des Programms mit RUN | RETURN |

Monitor: ? 10,20

30

41000

READY

C

Das BASIC-Schlüsselwort LET wird benutzt, um einer Variablen einen Wert zuzuweisen. Diese Zuweisung kann über die Tastatur erfolgen oder aber auch durch die Berechnung einer arithmetischen Formel, deren Ergebnis in der Variablen gespeichert wird. Das Schlüsselwort LET hat jedoch seine Bedeutung verloren, d.h. **es ist optional und kann weggelassen werden.** Diejenige Variable, die den Wert empfängt, ist stets links vom Gleichheitszeichen und der zuweisende Wert oder die Formel immer rechts vom Gleichheitszeichen.

Zu beachten ist, daß das Gleichheitszeichen in diesem Fall keine mathematische Gleichheit darstellt, es ist vielmehr das Symbol für die Wertezuweisung, das bedeutet, daß beide Seiten vom Gleichheitszeichen nicht unbedingt gleich groß sein müssen.

Das Berechnen von Formeln erfordert, daß in Rechenoperatoren die jeweiligen Anweisungen bestimmte Rangfolgen haben. In Klammern gesetzte Operationen werden zuerst ausgeführt und haben damit den höchsten Rang; es folgt das Potenzieren, rechnen mit negativen Zahlen, multiplizieren, dividieren, addieren und subtrahieren.

Es kann durch das Setzen von Klammern die Rangfolge geändert werden.

Damit gilt folgende Priorität (vgl. S. 41)

1. Stehen in einer Berechnung mehrere Klammern, so erfolgt zuerst die Berechnung der Klammern, und zwar von innen nach außen.

2. Der Commodore prüft auf Negativzahlen.

3. Sämtliche Exponentialrechnungen werden aufgeführt.

4. Es werden alle Multiplikationen und Divisionen von links nach rechts aufgeführt.

5. Alle Additionen und Subtraktionen werden von links nach rechts ausgeführt.

5.2 Vergleichsoperatoren

Bei den BASIC-Schlüsselworten in Kapitel 4 haben wir bereits gesehen, daß logische Vergleiche notwendig sind, um bedingte Sprünge ausführen zu können. Diese Bedingungen müssen vom Programmierer vorher definiert werden, vom Computer werden sie geprüft und als erfüllt anerkannt.

Die für BASIC festgesetzten Zeichen als Vergleichsoperatoren lauten:

A	=	B	Wenn A gleich B ist, dann ....
A	<>	B	Wenn A ungleich B ist, dann ....
A	>	B	Wenn A größer B ist, dann ....
A	> =	B	Wenn A größer oder gleich B ist, dann ....
A	<	B	Wenn A kleiner B ist, dann ....
A	< =	B	Wenn A kleiner oder gleich B ist, dann ....

Abb. 15: Vergleichsoperatoren

Grundsätzlich kann vor und hinter dem Vergleichsoperator entweder eine Variable, eine Konstante oder eine Formel oder ein Rechenausdruck stehen.

Weitere logische Operatoren sind die Worte AND, OR, NOT.

Sie entsprechen dem: AND logischen „und"
OR logisch „oder"
NOT logisch „nicht".

Diese logischen Operatoren haben in der Rangfolge der Ausführung die niedrigste Wertung, d.h. die Reihenfolge lautet:

1. Rechenoperatoren

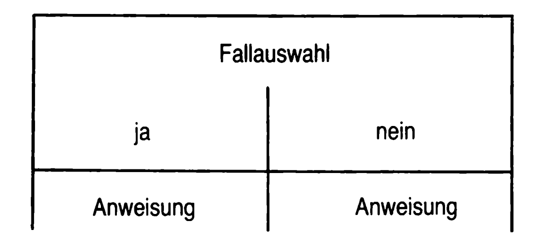
2. Vergleichsoperatoren

3. Logische Operatoren.

5.3 Der berechnete Sprung

BASIC-Schlüsselwort: IF ... THEN ...
oder
IF ... THEN ... ELSE ...

Struktursymbol:



5.3.1 Einseitige Auswahlstruktur

BASIC-Schlüsselwort: IF ... THEN ...

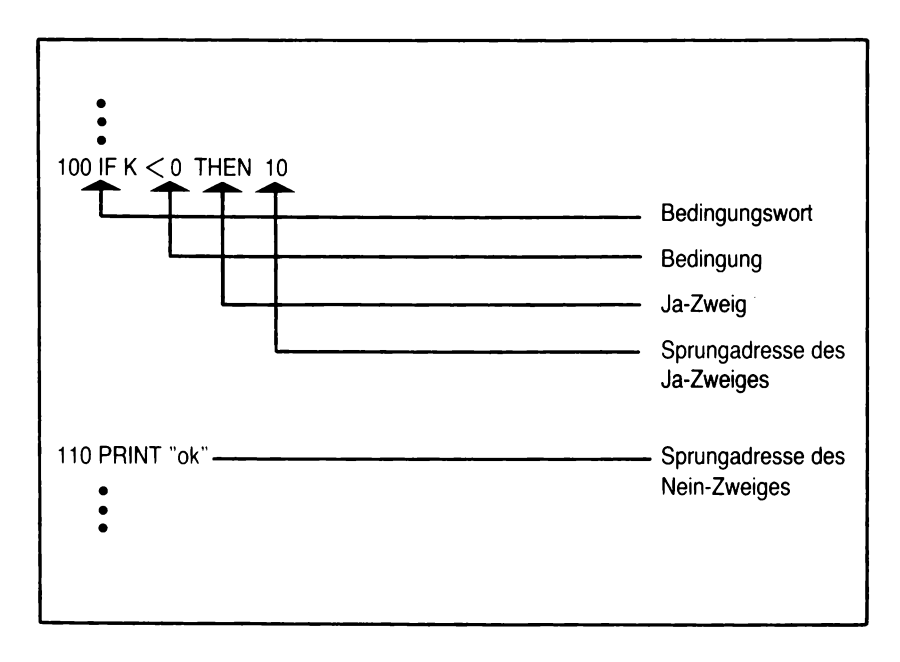
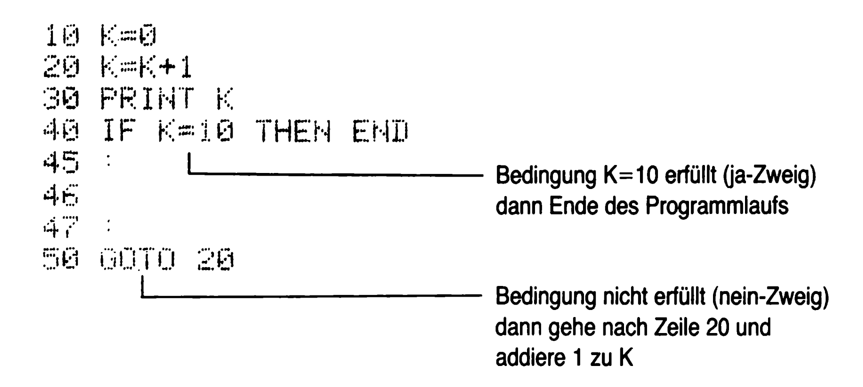


Abb. 16: Die IF ... THEN ... Anweisung

Beispiel: Im folgenden Programm wird bis zur Zahl 10 gezählt und die jeweils errechneten Werte einzeln auf dem Bildschirm ausgegeben.



Über das Schlüsselwort IF ... THEN ... wird, in Abhängigkeit von bestimmten Bedingungen, entschieden, welche Aktion ausgeführt werden soll. Aktionen, die ausgeführt werden sollen, können beispielsweise Sprünge in vorgegebene Zeilen, aber auch Anweisungen gemäß BASIC-Schlüsselworten, wie PRINT, INPUT u.a. sein.

Ist eine Bedingung erfüllt, dann wird das Programm entsprechend der THEN-Anweisung ausgeführt. Ist die Bedingung nicht erfüllt, dann wird das Programm in der nächsten Zeile fortgesetzt. Damit kann für die „einseitige Auswahl" folgendes formuliert werden:

„Wenn ...., dann tue dies, sonst aber tue nichts".

5.3.2 Zweiseitige Auswahlstruktur

BASIC-Schlüsselwort: IF ... THEN ... ELSE ...

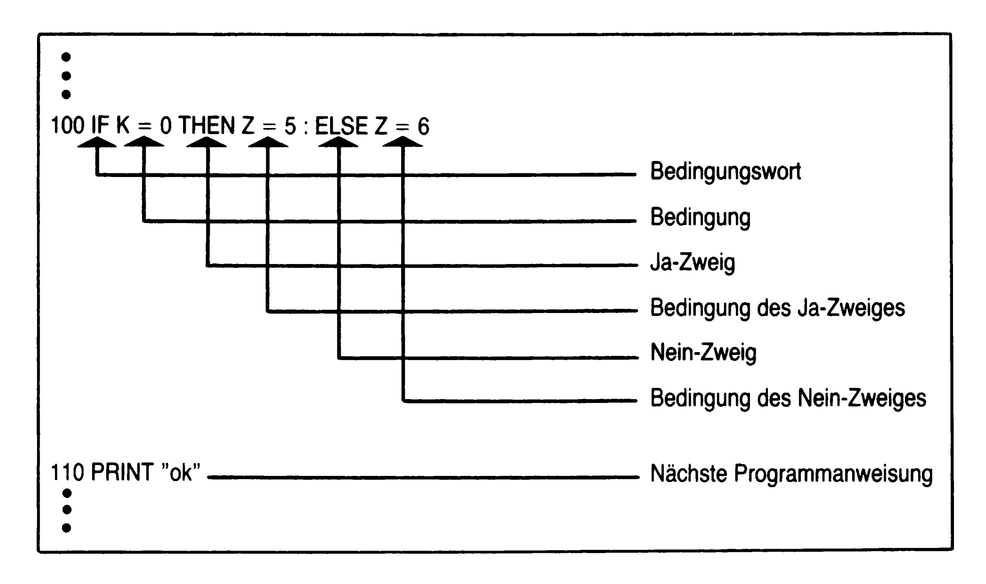


Abb. 17: Die IF ... THEN ... ELSE ... Anweisung

Beispiel: Es sind unterschiedliche Zinsen zu berechnen. Ist K kleiner als 1000, so sind 10 % zu berechnen, bei größer oder gleich 1000 sind 20 % zu berechnen.

```
10 INPUT"KAPITAL";K
20 IF K<1000 THEN P=10:ELSE P=20
30 :
40 :
50 :
60 :
70 :
80 Z=K/100*P
90 PRINT Z
```

Vergleichsoperator

Ja-Zweig
(P erhält den Wert 10)

Nein-Zweig
(P erhält den Wert 20)

In der zweiseitigen Auswahl wird das Schlüsselwort IF ... THEN ... durch das Wort ELSE ergänzt. **Die Bedingung IF THEN ist eine „wenn/dann"-Klausel; die Ergänzung um das Schlüsselwort ELSE ist eine „oder/dann"-Klausel.**

Ist eine Bedingung erfüllt, so wird das Programm entsprechend der THEN-Anweisung ausgeführt; ist die Bedingung nicht erfüllt, so wird sie entsprechend der ELSE-Anweisung ausgeführt.

HINWEIS: Beachten Sie, daß vor dem Wort ELSE unbedingt ein Doppelpunkt stehen muß.

Man kann damit die IF-THEN-ELSE-Konstruktion als zweiseitige Auswahl wie folgt formulieren:

„Wenn ..., dann tue dies, sonst aber tue das".

5.4 BEISPIEL: „Einfache Zinsrechnung" mit Negativ-Kontrolle und unterschiedlichen Zinssätzen

Es kann vorkommen, daß aus Versehen ein negativer Betrag als Kapital eingegeben wird. Der Computer berechnet dann negative Zinsen, obwohl dies ökonomisch sinnlos ist.

Zusätzlich hat sich unsere Bank entschlossen, bei Einlagen über 100.000,– DM 6 % Zinsen zu zahlen; darunter 5 %.

Problemanalyse:

Wir müssen negative Kapitalien abfragen durch eine IF ... THEN ... Abfrage und eine Meldung ausgeben, daß der Betrag falsch war. Eine Wiederholung der Eingabe muß möglich sein.

Das Problem der unterschiedlichen Zinsen lösen wir durch eine IF ... THEN ... ELSE ... Abfrage.

Variable:

Zusätzliche Variable benötigen wir nicht, so daß wir das Programm, das wir bisher hatten, ergänzen können.

STRUKTOGRAMM:

BILDSCHIRM FREI / ÜBERSCHRIFT

EINGABE ANZAHL KAPITALIEN N

WIEDERHOLE BIS N

EINGABE KØ

IST KØ POSITIV ?

JA	NEIN
EINGABE LAUFZEIT T	FEHLER

IST KØ KLEINER / GLEICH 10000

JA	NEIN
ZINSSATZ P = 5	ZINSSATZ P = 6

BERECHNE Z UND KN

AUSGABEN KN UND Z

ENDE

CODIERUNG:

```
10 PRINT "[illegible]"
20 PRINT "EINFACHE ZINSRECHNUNG"
30 PRINT
40 INPUT "WIEVELE KAPITALIEN?";N
50 PRINT
60 FOR I=1 TO N
70 INPUT "KAPITAL :";K0
80 IF K0<=0 THEN PRINT "FALSCHE EINGABE":GOTO 70
90 INPUT "LAUFZEIT:";T
100 IF K0<=10000 THEN P=5 :ELSE P=6
110 Z=K0*P*T/(360*100)
120 KN=K0+Z
125 PRINT
130 PRINT "NEUES KAPITAL =",KN
140 PRINT "ZINSEN        =",Z
145 PRINT
150 NEXT I
160 END
```

5.5 Kontrollfragen

1. Führen Sie das Programm „Rechnen mit 2 Zahlen" aus!

2. Was sind Vergleichsoperatoren und welche Bedeutung haben sie?

3. Erläutern Sie sich die Begriffe:
   - einseitige Auswahlstruktur
   - zweiseitige Auswahlstruktur.

4. Schreiben Sie ein Programm, das folgendes Problem löst:
   - Ein Schachbrett hat 64 Felder,
   - auf Feld 1 wird ein Getreidekorn gelegt, auf Feld 2 zwei Körner, auf Feld 3 vier Körner usw.,
   - also auf jedes Feld die doppelte Anzahl des vorhergegangenen Feldes.
   - Wieviel Körner liegen auf Feld 64?

5. Codieren Sie das folgende Struktogramm!

HINWEIS: Es ist ein Programm zum Raten von Zahlen; ein Spieler gibt die Zahl ein, sie wird gespeichert und der zweite Spieler soll sie erraten. Je niedriger die Anzahl der Rateversuche, umso besser!

```
BILDSCHIRM FREI / ÜBERSCHRIFT "ZAHLENARTEN"
AUSGABE DER SPIELANLEITUNG
    EINGABE DER RATEZAHL "A"
    BILDSCHIRM FREI
        EINGABE DER GERATENEN ZAHL "B"
        ZÄHLER UM 1 ERHÖHEN "Z=Z+1"
        IST DIE GERATENE ZAHL GLEICH DER GESUCHTEN ZAHL? "A=B"
          NEIN:
            IST A < B
              JA:   PRINT "ZAHL ZU GROSS"
              NEIN: PRINT "ZAHL ZU KLEIN"
          JA:
            PRINT "RICHTIG"
            PRINT "ZAHL DER VERSUCHE" Z
    NOCH EIN SPIEL ?
      JA
      NEIN
                 ENDE
```

Für Ihre Lösungen:

6. Weitere Eingabeanweisungen

Lernziel: Der Leser soll Warteschleifen programmieren und anwenden können sowie Daten im Programm selbst unterbringen können.

BASIC-Schlüsselwort: GET

READ

DATE

RESTORE

6.1 Warteschleifen

BASIC-Schlüsselwort: GET

Struktursymbol:

Beispiel:

```
10 PRINT "WARTESCHLEIFE! WENN WEITER BITTE 'W' DRUECKEN"
20 GET Y$
30 :
40 :
50 IF Y$="W" THEN 120
60 :
70 :
80 IF Y$<>"W" THEN 20
90 :
100
110 :
120 PRINT "O.K."

READY.
```

GET — Schlüsselwort für eine Taste
Y$ — Variablenname für die Taste
= — Vergleich auf „Gleichheit"
"W" — Taste "W" ist gefordert
<> — Vergleich auf Ungleichheit und Rücksprung solange, bis durch Gleichheit die Schleife verlassen werden kann

Oftmals ist es erforderlich, ein Programm anzuhalten, ohne seinen Ablauf abzubrechen. In einem solchen Falle bedient man sich einer Warteschleife, die über das BASIC-Schlüsselwort GET programmiert werden kann. **Die GET-Anweisung erwartet ein Zeichen als Tastatur-Eingabe, ohne daß die RETURN-Taste gedrückt werden muß.**

Erreicht ein Programm während das Abarbeitens eine GET-Anweisung, so muß ein Zeichen über die Tastatur eingegeben werden. Der GET-Anweisung hat eine Variable zu folgen, der dann ein bestimmter Wert oder ein Buchstabe zugewiesen wird. Solange keine Tastatureingabe erfolgt, verweilt das Programm in dieser Schleife.

Eine ähnliche Wirkung hat das Schlüsselwort GET .., hier wartet das Programm einfach, bis eine Taste gedrückt wird, ohne daß es sich um eine besondere Taste handeln muß. Mit jedem Tastendruck läuft das Programm weiter.

6.2 Daten im Programm

Lernziel: Der Leser soll erkennen, daß Daten auch im Programm selbst eingebaut und abgearbeitet werden können.

BASIC-Schlüsselwort: DATA

READ

RESTORE

Struktursymbol:

Anweisung

Beispiel:

```
10 DATA 10,20,30
20 :
30 :
40 :
50 FOR I=1 TO 3
60 READ D
70 :
80 :
90 :
100 PRINT D
110 NEXT I
120 RESTORE
```

a) DATA

Beim Programmablauf müssen den Variablen Werte zugewiesen werden. Dies ist, wie wir sehen, möglich über die BASIC-Schlüsselworte INPUT oder GET. Ebenfalls ist es möglich, Daten aus den Peripheriegeräten, d.h. Cassette oder Floppydisk abzurufen.

Eine andere Form, Daten in die Variablen einzugeben, ist das BASIC-Schlüsselwort DATA. Diese **Datenspeicherung innerhalb eines Programms** wird dann benutzt, wenn es sich nicht um all zu viele Daten handelt und die Datenspeicherung auf einer Cassette oder Floppydisk nicht komfortabel ist.

Das BASIC-Schlüsselwort DATA wird in das Programm eingefügt; der DATA-Anweisung folgt eine Liste von Elementen, die durch einen gesonderten Befehl (READ) später eingelesen werden. Die Elemente können Zahlen oder STRINGS sein und müssen durch Kommata getrennt werden. Die STRINGS benötigen keine Anführungszeichen, lediglich in einigen Sonderfällen sind Anführungszeichen zuzufügen, und zwar dann, wenn in den STRINGS Leerstellen, Doppelpunkte oder Kommata enthalten sind.

b) READ

Um die mit dem BASIC-Schlüsselwort DATA gespeicherten Daten abzurufen, benötigt man das BASIC-Schlüsselwort READ. Dabei werden die Daten der Reihe nach, d.h. beginnend bei dem ersten Wert der hinter dem Schlüsselwert DATA steht, ausgerufen. Intern setzt das Gerät einen sogenannten "POINTER", der sicherstellt, daß der jeweilig nächstfolgende DATA-Wert auch tatsächlich abgerufen wird.

Diese DATA-Werte werden in dafür vorgesehene Variable aufgenommen und können über diese Variable abgearbeitet werden.

Da die DATA-Zeilen in die Variablen aufgerufen werden, ist zu beachten, daß die Wertezuweisungen auch entsprechend der jeweiligen Variablen erfolgen. Dies bedeutet, daß in eine Variable, die als Rechenvariable ohne Zusatzsymbol (Prozent oder Dollar) enthalten ist, auch keine STRINGS eingelesen werden dürfen. Werden STRINGS falscherweise eingelesen, so führt dies zu einer Fehlermeldung!

Eine interessante Form der Abarbeitung einer DATA-Anweisung besteht in Verbindung mit der FOR ... TO ...NEXT-Schleife.

c) RESTORE

Sollen Daten, die in einem Programm mit dem BASIC-Schlüsselwort DATA abgespeichert wurden, mehrfach abgearbeitet werden, so ist der "POINTER" auf den ersten Wert nach der DATA-Anweisung zurückzusetzen. Dies geschieht mit Hilfe des BASIC-Schlüsselworts RESTORE. Damit ist es möglich, DATA-Zeilen mehrfach zu lesen.

Benutzt man die RESTORE-Anweisung ohne Zeilennummer, so wird der "POINTER" auf die allererste Anweisung nach dem Schlüsselwort DATA im Programm zurückgesetzt. Folgt der RESTORE-Anweisung eine Zeilennummer, dann wird der "POINTER" auf den ersten Wert, der nach der DATA-Anweisung folgt, in derjenigen Zeile gesetzt, die nach RESTORE genannt wurde.

6.3 BEISPIEL: „Einfache Zinsrechnung" mit Daten im Programm

Es stört, daß wir die Kunden immer fragen müssen, wieviele Kapitalien zu berechnen sind. Das Programm soll beliebig oft durchlaufen werden können.

Gleichzeitig bieten wir unseren Kunden eine alternative Berechnung an; die Laufzeit wird für 180 Tage und für 360 Tage festgelegt.

Problemanalyse:

Durch das Programmieren einer Warteschleife können wir erreichen, daß das Programm nach Wunsch abgebrochen werden kann. Als Befehlsfolge benutzen wir eine GET Anweisung.

Die alternative Laufzeit von 180 bzw. 360 Tagen fügen wir mit DATA fest im Programm ein und lassen die Berechnung über eine FOR ... TO ... Schleife zweimal erfolgen. Mit READ rufen wir die jeweiligen Werte für die Laufzeit ab und setzen den Pointer nach der NEXT Anweisung zurück.

Variable:

Als neue Variable benötigen wir:

Y = Variable zur GET-Anweisung

BILDSCHIRM FREI / ÜBERSCHRIFT

FESTLEGEN DER DATEN

EINGABE KØ

WIEDERHOLE ZWEIMAL

IST KØ NEGATIV ?
NEIN | JA

LESEN DATEN | FEHLER

IST KØ KLEINER / GLEICH 10000
JA | NEIN

ZINSSATZ P = 5 | ZINSSATZ P = 6

BERECHNE Z UND KN

AUSGABEN KN UND Z

ZURÜCKSETZEN POINTER

NOCH EINE BERECHNUNG
JA | NEIN

ENDE

CODIERUNG:

```
10 PRINT "□"
20 PRINT "EINFACHE ZINSRECHNUNG"
25 PRINT
30 DATA 180,360
40 INPUT "KAPITAL :";K0
50 PRINT
60 FOR I=1 TO 2
70 PRINT
80 IF K0<=0 THEN PRINT "FALSCHE EINGABE":GOTO 40
90 READ T
100 IF K0<=10000 THEN P=5 :ELSE P=6
110 Z=K0*P*T/(360*100)
120 KN=K0+Z
125 PRINT
130 PRINT "NEUES KAPITAL BEI";T;"TAGEN =",KN;"DM"
140 PRINT "ZINSEN          =",Z;"DM"
145 PRINT
150 NEXT I
160 RESTORE
170 PRINT
180 PRINT"NOCH EINE BERECHNUNG? J/N"
190 GET Y$
200 IF Y$="J" THEN PRINT"□":GOTO 30
210 IF Y$="N" THEN 230
220 GOTO 190
230 END
```

6.4 Kontrollfragen

1. Welche Bedeutung haben Warteschleifen?

2. Welche Möglichkeiten gibt es, um Werte in das Programm einzugeben?

3. Wie unterscheidet sich DATA von INPUT und GET?

4. Die Mittel- und Tiefsttemperaturen von Mainz betrugen 15.2. rund -28; die von Wiesbaden 15.6. rund -27.
   Schreiben Sie ein Programm, das diese Daten enthält und zeilenweise ausgibt!

Für Ihre Lösungen:

7. Weitere Strukturanweisungen

Lernziel: Der Leser soll weitere Anweisungen, die das Strukturieren von BASIC-Programmen ermöglichen, erlernen und anwenden können und zusätzliche BASIC-Befehlsworte anwenden können.

BASIC-Schlüsselwort: ON ... GOTO ...

GOSUB ... RETURN

ON ... GOSUB

STOP

END

REM

Die BASIC Schlüsselworte END, REM und STOP dienen dazu, ein Programm weiter zu strukturieren. Dabei übernehmen Sie überwiegend Funktionen der Lesbarkeit bzw. Fehlerkorrektur von Programmen.

END = Sollte am Ende eines jeden Programmes stehen; ein Programm kann allerdings so aufgebaut sein, daß das Ende der Arbeit in einer vorgelagerten Zeilennummer steht (siehe Bsp. 7.2).

REM = Dieses Schlüsselwort beeinflußt den Ablauf eines Programmes nicht, d.h. es ist damit keine echte Strukturanweisung. Wird jedoch ein Programm ausgelistet, so erscheinen auch die REM-Zeilen, die also dafür genutzt werden können, um ein Programm lesbarer zu gestalten (siehe Bsp. 7.2).

STOP = Der Ablauf eines Programmes wird hier angehalten und kann mit CONT fortgesetzt werden, ohne daß die Inhalte von Variablen zerstört werden (siehe Kapitel „Fehlersuche").

7.1 Fallabfragen

BASIC-Schlüsselwort: ON .. GOTO ...

Struktursymbol:

Fallabfrage			
Weg 1	Weg 2	Weg 3	usw.

```
5 PRINT "J"
10 PRINT "WAS WOLLEN SIE TUN?"
20 PRINT "BITTE AUSWAEHLEN UND DIE"
30 PRINT "ZUGEHOERIGE NUMMER EINGEBEN!"
40 PRINT:PRINT
50 PRINT "LESEN     = 1"
60 PRINT "SCHREIBEN= 2"
70 PRINT "NICHTS    = 3"
75 :
76 :
80 PRINT:PRINT
90 INPUT "IHRE ZAHL?";A
95 :
96 :
100 PRINT:PRINT
110 ON A GOTO 120,130,140
112 :
113 :
114 :
115 :
116 :
117 :
118 :
120 PRINT "BUCH KAUFEN!":END
130 PRINT "PAPIER UND KULI HOLEN!":END
140 PRINT "INS BETT GEHEN!":END
```

Mögliche Fälle

Variable für den ausgewählten Fall

Schlüsselwort

Sprungadresse für A = 1

Sprungadresse für A = 2

Sprungadresse für A = 3

Zur Schachtelung von Fallabfragen haben wir bisher das BASIC-Schlüsselwort IF ... THEN benutzt. Diese Schachtelung wird unübersichtlich, wenn mehr als zwei Auswahlstrukturen vorgegeben werden. **Zur Vereinfachung der mehrseitigen Auswahl benutzt man deshalb die Fallabfrage mit dem BASIC-Schlüsselwort ON ... GOTO.**

Dem Wort ON hat zuerst eine Variable zu folgen, der ein Wert, beginnend ab einschließlich eins, zugewiesen wird. Dem Wort GOTO werden Zeilennummern angefügt, in die verzweigt werden soll. Hat die Variable den Wert eins, so wird in diejenige Zeilennummer verzweigt, die als erstes hinter dem BASIC-Schlüsselwort GOTO steht. Hat die Variable den Wert zwei, so wird in diejenige Zeile verzweigt, die als zweites hinter dem BASIC-Schlüsselwort GOTO steht usw.

Es ist zu beachten, daß die Anweisung ON ... GOTO grundsätzlich ganzzahlige Werte erwartet. Eingabefehler müssen im Programm selbst abgefangen werden. Wird eine negative Zahl eingegeben, so gibt das System eine Fehlermeldung aus.

7.2 Unterprogramme

BASIC-Schlüsselwort: GOSUB

Struktursymbol:

Anweisung (GOSUB)

Beispiel:

```
10 REM EINGABE IM SUBPROGRAMM
20 GOSUB 1000
30 :                     Schlüsselwort
40 :
50 :                     Sprungadresse
60 PRINT A,B
70 END
1000 INPUT A,B
1010 RETURN
                         Rückkehr nach Zeile 30

READY.
```

Werden innerhalb eines Programmes bestimmte Abläufe, Berechnungen u.a. mehrfach benötigt, so müssen sie an und für sich an jeder Stelle erneut programmiert werden. Diese gleichartige Arbeit kann durch die Unterprogrammtechnik erleichtert werden, denn der Ablauf oder die Berechnung können in einem Unterprogramm untergebracht werden. Dieses Unterprogramm wird immer dann aufgerufen, wenn es im Hauptprogramm benötigt wird. Damit erspart man sich die Arbeit, ein und dieselbe Situation mehrfach zu programmieren.

Unterprogramme sind also Programme, die öfter gebrauchte Teile des Hauptprogramms beinhalten und die dann beliebig oft innerhalb des Gesamtprogramms aufgerufen werden können.

Unterprogramme bieten entscheidende Vorteile:

1. Ein an mehreren Stellen im Programm benötigter Ablauf muß nur einmal programmiert werden.

2. Oft benötigte Verfahren können auch in neuen Programmen eingesetzt werden.

3. Ein in Unterprogrammen gegliedertes Programm ist besser lesbar und verstehbar als ein ungegliedertes Gesamtprogramm.

Unterprogramme werden durch das BASIC-Schlüsselwort GOSUB, dem die Zeilennummer folgt in die verzweigt werden soll, aufgerufen. Die Zeilennummer, die aufgerufen wird, muß den ersten Befehl des betreffenden Unterprogramms enthalten. Der Computer setzt intern einen „POINTER" und merkt sich damit die, dem GOSUB-Schlüsselwort folgende, Zeilennummer; wenn das Unterprogramm abgearbeitet ist, muß der Beendigungsbefehl RETURN erfolgen. Das Wort RETURN bewirkt, daß das Unterprogramm an diejenige Stelle im Hauptprogramm zurückspringt, an der der „POINTER" steht. Dies ist die dem GOSUB-Befehl folgende Zeile.

Ein Unterprogramm wird auch als Subroutine bezeichnet.

Eine Sonderform der Subroutinen ist das BASIC-Schlüsselwort ON ... GOSUB. Dem Schlüsselwort ON folgt eine Variable, die die Werte ab einschließlich eins enthalten kann. Die Sprungfolge ist dieselbe wie bei dem Schlüsselwort ON ... GOTO.

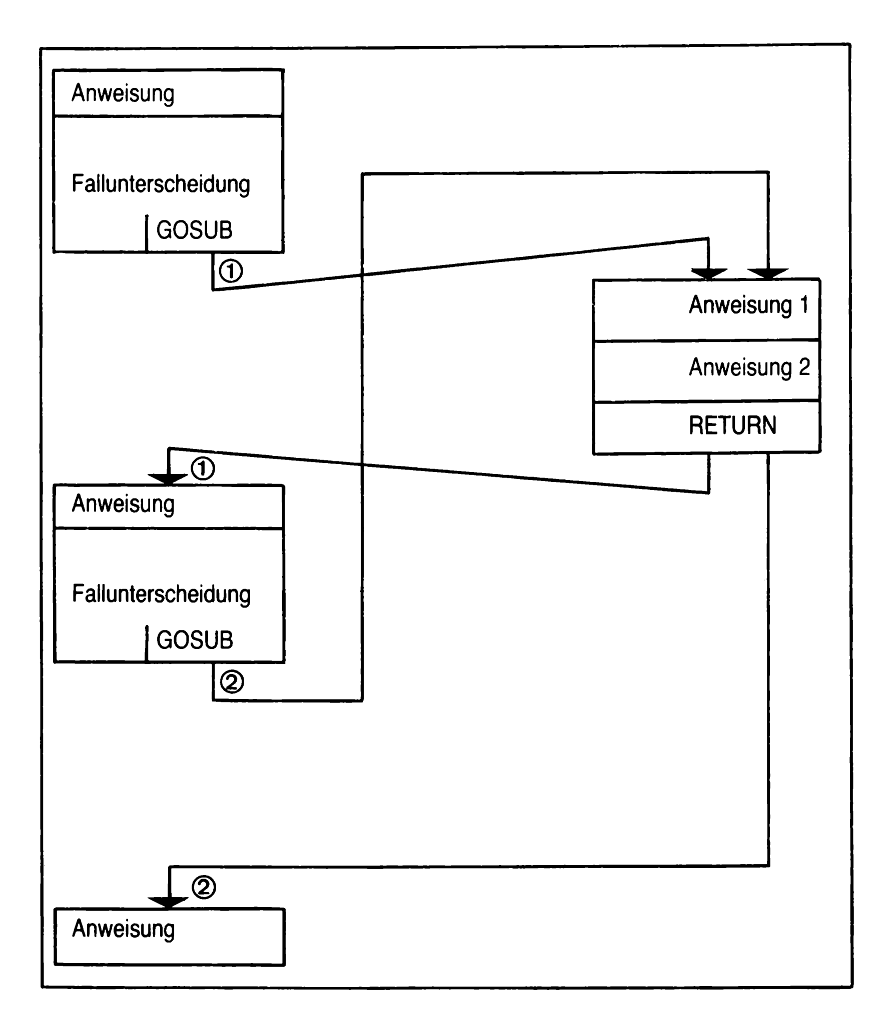


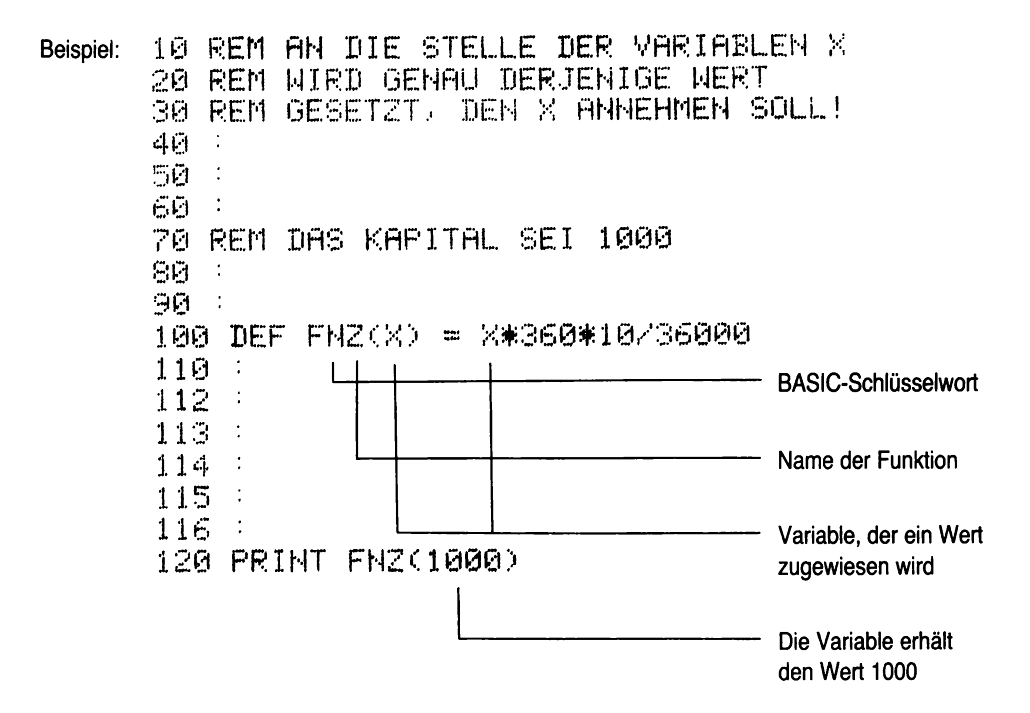
Abb. 18: Unterprogramme

7.3 Standardfunktionen

Über das BASIC-Schlüsselwort DEF FN können Funktionen mit einem bestimmten Namen versehen werden. Diese Funktionen kann man als besondere Unterprogramme betrachten, die stets mit ihrem Namen aufgerufen werden. Häufig wiederkehrende Probleme der Mathematik sind in Funktionen standardmäßig vorgegeben (vgl. hierzu die entsprechenden Seiten im Bedienungshandbuch), aber für spezielle Benutzerprobleme können sie selbst definiert werden.

Der Vorteil des BASIC-Schlüsselwort DEF FN liegt darin, daß sich eine komplexe Funktion mit einem kurzen Namen (Variable) bezeichnen läßt. Der Vorteil liegt darin, daß Speicherplatz gespart wird, aber auch, daß Fehler bei der Eingabe der Funktion vermieden werden.

Der Name, den die Funktion erhält, muß stets mit FN beginnen.



7.4 BEISPIEL: Einfache Zinsrechung und Zinseszinsrechnung mit Fallabfrage

Es ist möglich, daß unsere Kunden ihre Kapitalien neben der einfachen Verzinsung auch auf Zinseszinsen anlegen wollen, d.h. neben der Zinsformel ist die Zinseszinsformel anzuwenden.

$$\text{Zinseszinsformel: } KN = KØ * \left(1 + \frac{P}{100}\right)^{J}$$

Problemanalyse:

Wir müssen dem Benutzer unseres Programmes die Möglichkeit bieten, die jeweils gewünschte Berechnung auswählen zu können. Dies erreichen wir durch eine Programmierung mit ON ... GOTO ..., die eine Auswahl ermöglicht.

Da die Beträge in DM gemessen werden, aber das Gerät auf mehrere Stellen nach dem Komma genau rechnet, müssen wir die „Kaufmännische Rundung" anwenden und programmieren.

Dies geschieht über ein Unterprogramm, das wir mit GOSUB aufrufen.

Variable:

Zusätzliche Variable ist:

J = Laufzeit des Anfangskapitals KØ in Jahren.

STRUKTOGRAMM:

BILDSCHIRM FREI / ÜBERSCHRIFT

FALLAUSWAHL

EINFACHE ZINSEN = 1

EINGABE KØ, P, T

EINGABE POSITIV

JA

NEIN

BERECHNE Z

RUNDEN "GOSUB 1000"

BERECHNEN KN

RUNDEN "GOSUB 1020"

AUSGABE KN, Z

ZINSESZINSEN = 2

EINGABE KØ, P, J

EINGABE POSITIV

JA

NEIN

BERECHNE KN

RUNDEN "GOSUB 1020"

BERECHNE Z

RUNDEN "GOSUB 1000"

AUSGABE KN, Z

ENDE = 3

ENDE

UNTERPROGRAMM 1000 RUNDEN Z

UNTERPROGRAMM 1020 RUNDEN KN

CODIERUNG:

```
5 PRINT"♥"
10 PRINT "EINFACHE UND ZINSESZINSEN"
20 PRINT:PRINT
30 PRINT "WAS SOLL BERECHNET WERDEN?"
40 PRINT "BITTE ZAHL EINGEBE!!!"
50 PRINT:PRINT
60 PRINT "EINFACHE ZINSEN = 1"
70 PRINT "ZINSESZINSEN    = 2"
80 PRINT "PROGRAMM BEENDEN= 3"
90 INPUT A
100 ON A GOTO 200,300,110
110 END
200 PRINT:PRINT
210 INPUT "ANFANGSKAPITAL    ".K0
220 INPUT "LAUFZEIT IN TAGEN";T
230 INPUT "ZINSSATZ          ";P
233 IF K0=<0 OR T=<0 OR P=<0 THEN PRINT"FALSCHE EINGABE":GOTO 210
235 PRINT:PRINT
240 Z=K0*P*T/(360*100)
245 GOSUB 1000
250 KN=K0+Z
255 GOSUB 1020
260 PRINT "NEUES KAPITAL =";KN;"DM"
270 PRINT "ZINSEN        =";Z;"DM"
280 PRINT:PRINT
290 GOTO 20
300 PRINT:PRINT
310 INPUT "ANFANGSKAPITAL     ;K0
320 INPUT "LAUFZEIT IN JAHREN";J
330 INPUT "ZINSSATZ           ";P
333 IF K0=<0 OR J=<0 OR P=<0 THEN PRINT"FALSCHE EINGABE":GOTO 310
335 PRINT:PRINT
340 KN=K0*(1+P/100)↑J
345 GOSUB 1020
350 Z=KN-K0
355 GOSUB 1000
360 PRINT "NEUES KAPITAL   =";KN;"DM"
370 PRINT "ZINSEN          =";Z;"DM"
380 PRINT:PRINT
390 GOTO 20
1000 Z=INT(Z*100+0.5)/100
1010 RETURN
1020 KN=INT(KN*100+0.5)/100
1030 RETURN

READY.
```

7.5 Kontrollfragen

1. Welche Bedeutung haben die BASIC-Schlüsselwort REM und STOP im Programm?
2. Erläutern Sie den Begriff „Fallabfrage"!
3. Wann benutzt man die Unterprogramm-Technik?
4. Wie kann man sich die kaufmännische Rundung vorstellen und wie ist sie zu programmieren?
5. Interpretieren Sie die Zeilen 233 und 333 im Beispiel „Einfache Zinsrechnung" auf Seite 90!
6. Welche Wirkung hat folgender Befehl (vgl. Bsp. S. 90):

```
5 PRINT " SHIFT + HOME/CLR "
```

und der Befehl (vgl. Bsp. S. 103):

```
5 SCNCLR
```

7. Entwickeln Sie ein Programm zur Prozentrechnung! Der Benutzer soll die Wahl haben, zwischen der Berechnung des:

$$\text{Prozentwertes} = \frac{\text{Grundwert} * \text{Prozentsatz}}{1ØØ}$$

$$\text{Grundwertes} = \frac{\text{Prozentwert} * 1ØØ}{\text{Prozentsatz}}$$

$$\text{Prozentsatzes} = \frac{\text{Prozentsatz} * 1ØØ}{\text{Grundwert}}$$

Für Ihre Lösungen:

8. Tabellenverarbeitung mit ergänzenden Ausgabeanweisungen

Lernziel: Der Leser soll die grundlegenden Befehle der Tabellenüberarbeitung kennen und anwenden lernen.

BASIC-Schlüsselworte: DIM

COLOR

GRAPHIC

DRAW

SCNCLR

8.1 Variablenfelder

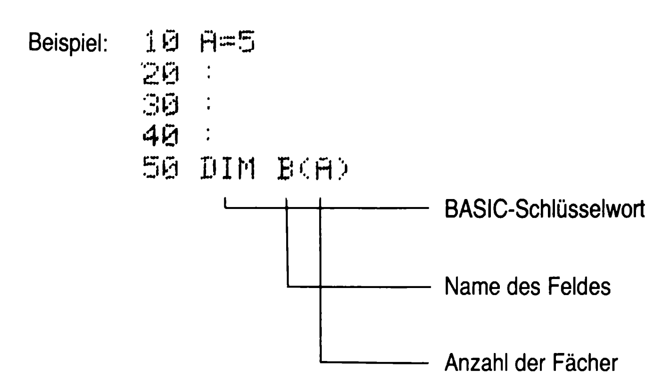
Mit dem BASIC-Schlüsselwort DIM (= Dimension) werden in der Zentraleinheit des Computers Speicherplätze reserviert. Diese Speicherplätze werden als Tabelle, ARREY, Bereich, Liste, Feld oder Matrix/Vektor bezeichnet.

Die Daten, die in dieses Feld geschrieben werden, können entweder INTEGER-Variable (%), Real-Variable (ohne Kennzeichen) oder STRING-Variable ($) sein.

Ein solches Feld oder ARREY kann man sich als „Regal" mit mehreren Schubfächern als Elemente vorstellen.

Das BASIC-Schlüsselwort DIM gibt dem Computer die Anweisung, ein Feld zu definieren. Der Name des Feldes ist ein Variablenname. Hinter dem Namen der Variablen muß in Klammern stehen, wie groß das Feld sein soll. Weil die DIM-Anweisung ein Feld festlegt, das später mit Werten aufgefüllt wird, ist sie eine Sonderform der Variablen, und zwar eine sogenannte Indexvariable. Sie sollte stets am Anfang eines Programmes stehen.

Mit der Anweisung DIM B (A) werden A + 1 Speicherplätze reserviert; also immer ein Speicherplatz mehr als in der Klammer angegeben wird.



8.2 Eindimensionale Tabellen

Eine **eindimensionale Tabelle** kann man sich als waagerecht oder als senkrecht angeordnet vorstellen. **Sie dehnt sich also immer in einer Richtung aus.**

Das Grundprinzip lautet wie folgt:

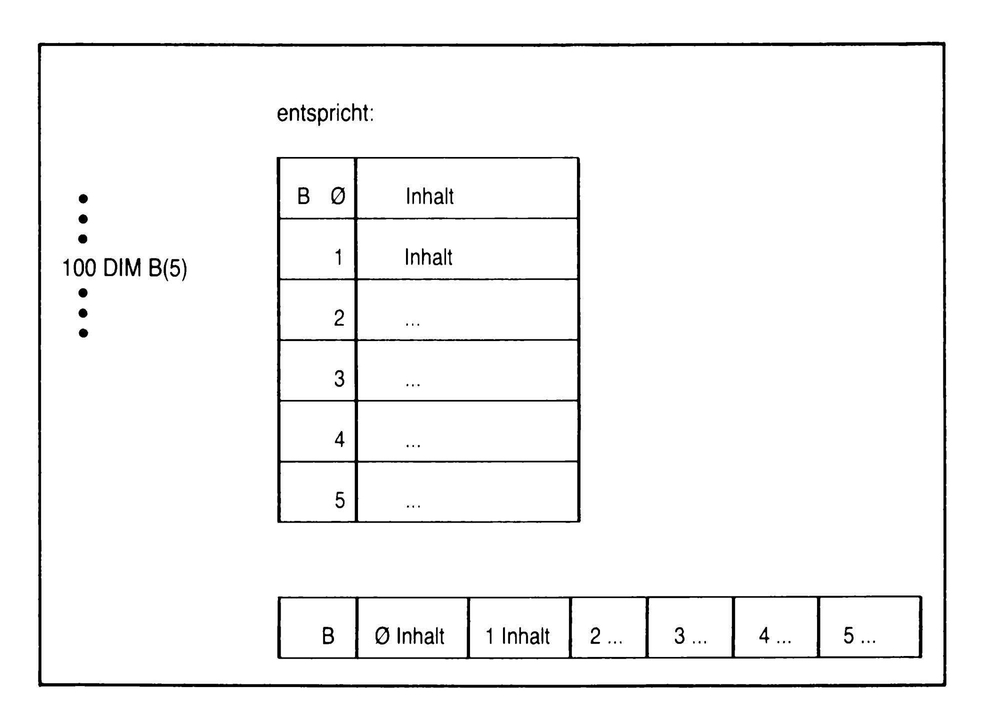


Abb. 19: Eindimensionale Tabellen

Dabei ist B der Name der Tabelle. Jedes einzelne Fach kann durch einen Index gefunden werden und der Inhalt über den Variablennamen verarbeitet werden.

Beispielsweise kann mit B (1) der Inhalt von Fach 1 abgerufen werden.

8.3 Zwei/Dreidimensionale Tabellen

Eine zweidimensionale Tabelle dehnt sich waagerecht und senkrecht aus. Damit kann man sich eine solche Tabelle als einen Regalschrank vorstellen, der eine bestimmte Anzahl von Fächern hat. Dabei wird wiederum durch das BASIC-Schlüsselwort DIM das Gerät angewiesen, in der Zentraleinheit einen solchen Regalschrank zu reservieren, der Name des Schrankes eine Variable und die Größe des Schrankes in Klammern angegeben wird. Bei einer zweidimensionalen Tabelle stehen in Klammern zwei Zahlen, die durch Komma getrennt sind, dabei gibt die erste Zahl die Anzahl der zu reservierenden Zeilen und die zweite Zahl die Anzahl der zu reservierenden Spalten an.

Die Anweisung DIM A (Z,S) reserviert damit Z+1 Zeilen und S+1 Spalten, also wie bei der eindimensionalen Tabelle, immer eine Stelle mehr als in Klammern angegeben ist.

Dabei gilt grundsätzlich folgendes:

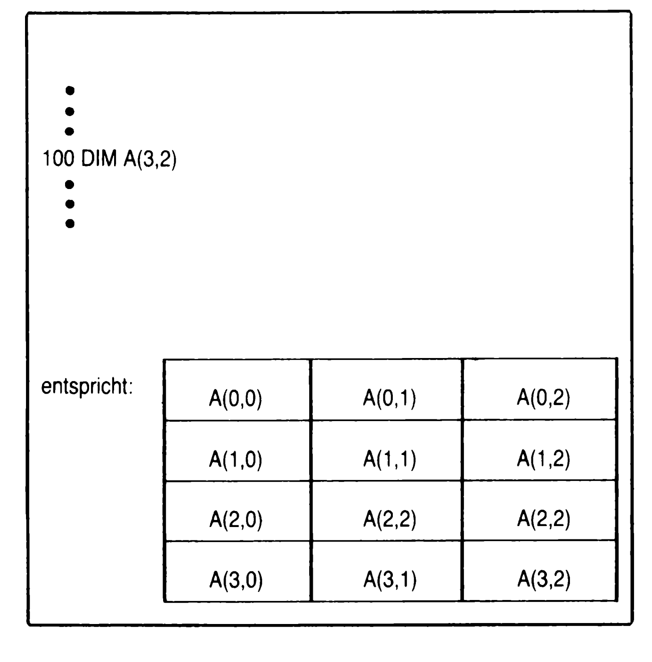


Abb. 20: Zweidimensionale Tabellen

Wobei A der Name des Schrankes ist und jedes Fach über die Zeilennummer und die Spaltennummer (den Index) gefunden werden kann.
Beispielsweise kann mit A (1,1) der Inhalt des Faches der zweiten Zeile und zweiten Spalte abgerufen werden.
Mit dem Commodore BASIC lassen sich neben ein- oder zweidimensionalen Tabellen auch dreidimensionale Tabellen erstellen. Das Grundprinzip ist das gleiche wie vorher beschrieben.

8.4 Eingabe und Ausgabe in Tabellen

Das BASIC-Schlüsselwort DIM und damit die Reservierung von Feldern kann sehr elegant verknüpft werden mit dem BASIC-Schlüsselwort FOR ... TO ... NEXT. Dabei wird die Tabellenbearbeitung häufig mit Dateibearbeitung kombiniert. Aus einer Datei (z.B. Kundenkartei) werden die Informationen in den Hauptspeicher eingelesen und in einer eindimensionalen oder zweidimensionalen Matrix abgelegt. Im Direktzugriff können diese Datensätze jetzt bequem abgearbeitet werden; Direktzugriff bedeutet, daß auf jedes Datenelement eines jeden Feldes über den Feldindex direkt zugegriffen werden kann. Nach der Bearbeitung können die Daten aus dem Feld wieder auf Cassette oder Diskette zurückgeschrieben werden.

Beispiel:

```
10 REM EINGABE UND AUSGABE IM VEKTOR
20 :
30 DIM A(5)
40 :
50 :
60 FOR I=1 TO 5
70 INPUT A(I)
80 :
90 :
100 NEXT I
110 FOR I=1 TO 5
120 PRINT I"TER WERT IST" SPC(2) A(I)
130 :
140
150 NEXT I
160 END
```

Festlegen des Feldes A mit 5+1 Fächern (zu Zeile 30: A(5))

Eingabe in Feld A (zu Zeile 70: A)

Auffüllen der Fächer I, denn I als Schleifenzähler wird immer um 1 hochgesetzt (zu Zeile 70: I)

Schleifenzähler (zu Zeile 120: I)

Für die DIM-Anweisung gelten noch einige Besonderheiten. So wird z.B. mit jedem Variablen-Namen automatisch ein eindimensionales Feld mit 11 Elementen angelegt. Dies bedeutet für das BASIC-Schlüsselwort DIM, daß es nur dann benutzt werden muß, wenn eindimensionale Felder mit mehr als 11 Elementen angelegt werden sollen. Zugleich kann man es aber auch benutzen, um Speicherplatz zu sparen, und zwar dann, wenn Felder mit weniger als 11 Elementen angelegt werden sollen.

Den Gesamtspeicherbedarf eines jeden Feldes kann man berechnen, indem man die Gesamtzahl der Elemente jeder Dimensionierung multipliziert. Zu beachten ist dabei, daß immer N+1 Felder festgelegt werden, d.h. jedes Feld startet bei Null. Der Speicherbedarf des Feldes A (3,2) ist damit 4 x 3, d.h. 12 Felder werden reserviert.

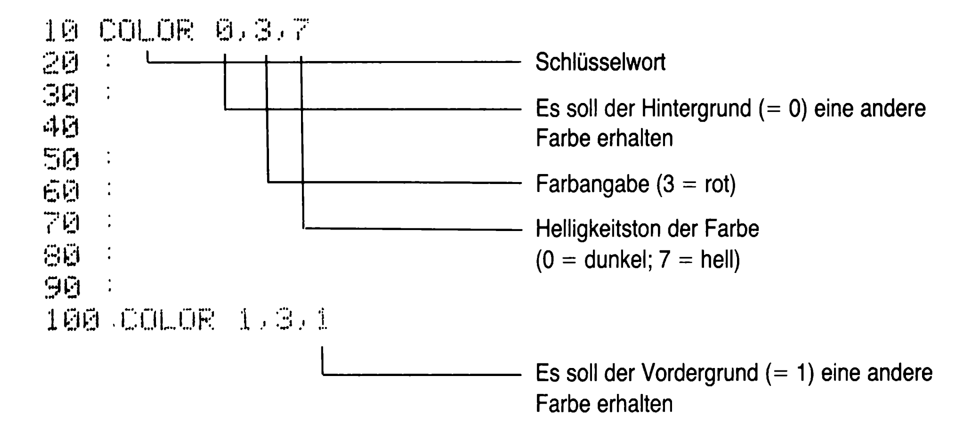
8.5 Zusätzliche Ausgabeanweisungen

Mit folgenden Befehlen kann die Ausgabe weiter gesteuert und verändert werden.

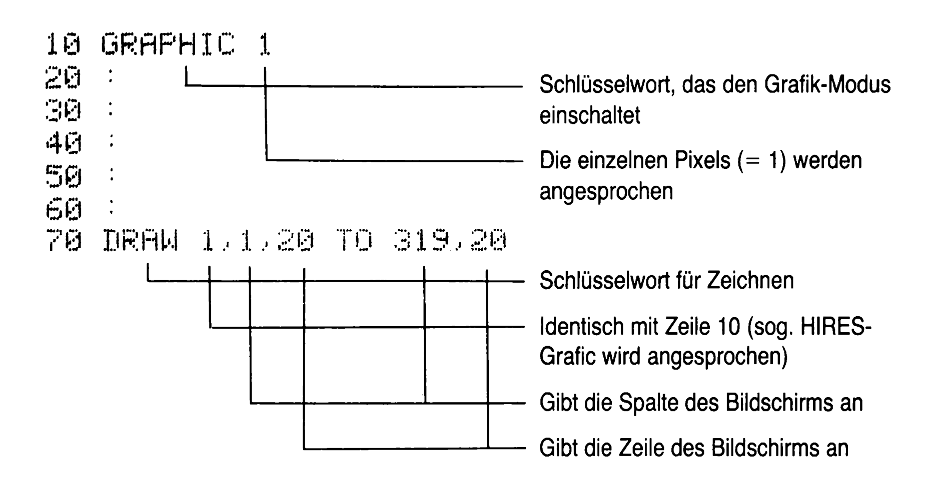
BASIC-Schlüsselwort: COLOR
GRAPHIC
DRAW
SCNCLR

Struktursymbol: Anweisung

Mit dem BASIC-Schlüsselwort kann die Farbe des Bildschirms verändert werden. Dabei ist auf die jeweilige Farbe umzuschalten und mit dem Befehelswort COLOR abzuarbeiten. Es gilt folgendes Grundprinzip (die Nummern der Farben bitte dem Handbuch entnehmen):



Das **BASIC-Schlüsselwort GRAPHIC schaltet auf den Grafik-Modus** des Gerätes um, d.h. es können in Zusammenhang mit dem Schlüsselwort DRAW oder auch CIRCLE bestimmte Zeichnungen auf den Bildschirm gebracht werden. Es gilt folgendes Grundprinzip (Einzelheiten wiederum dem Handbuch entnehmen):



Im obigen Beispiel wird ein Strich von der Spalte 1 bis zu Spalte 319 in der Zeile 20 gezogen. Der Bildschirm kann mit der Taste RUN/STOP + SHIFT verlassen werden.

Mit dem **BASIC-Schlüsselwort SCNCLR kann der Bildschirm frei** gemacht werden; der Cursor springt in die rechte obere Ecke zurück.

Bisher hatten wir für das Freimachen des Bildschirms die Befehlsfolge

```
   .
   .
   .
Anw. Nr. PRINT " SHIFT + CLEAR/HOME "
   .
```

benutzt.

Diese Anweisungen löschen den Inhalt der Variablen nicht.

8.6 BEISPIEL: Einfache Zinsrechnung in Tabellenverarbeitung

Bisher mußten wir alle Kapitalien einzeln eingeben und die Berechnung des jeweiligen Kapitals abwarten, bevor das nächste eingegeben werden konnte.

Jetzt soll es möglich sein, alle Kapitalien hintereinander einzugeben; alle sollen hintereinander berechnet und die Ergebnisse abgespeichert werden und anschließend sollen alle hintereinander ausgegeben werden.

Problemanalyse:

Um alle Kapitalien und Ergebnisse aufnehmen zu können, müssen wir Felder mit DIM reservieren. Zur Vereinfachung nehmen wir in der Zinsformel einen festen Zinssatz von 10 % an; er wird als Konstante in die Formel aufgenommen. Die Ausgabe soll in anderer Farbe auf dem Bildschirm erscheinen; wir erreichen dies mit COLOR.

Die Eingabe, Berechnung und Ausgabe schließen wir an den Zähler einer FOR ... TO-Schleife an. Zugleich soll die Forderung nach weiterer Berechnung über eine Warteschleife mit GET erfüllt werden.

Variable:

Da wir alle Werte in ein Feld aufnehmen, verlieren alle bisher gültigen Namen ihre Bedeutung; wir müssen die Feldbezeichnung in die Formel aufnehmen:

N	=	Anzahl der Kapitalien
A	=	Name des Feldes
I	=	Schleifenzähler
A(I,1)	=	Anfangskapital
A(I,2)	=	Laufzeit in Tagen
A(I,3)	=	Zinsen
A(I,4)	=	Neues Kapital
P	=	Warteschleife

STRUKTOGRAMM:

BILDSCHIRM FREI / ÜBERSCHRIFT

EINGABE ANZAHL KAPITALIEN

FELDRESERVIERUNG MIT DIM A(N,4)

WIEDERHOLE BIS N

- EINGABE ANFANGSKAPITAL A(I,1)
- EINGABE POSITIV?
  - JA
    - EINGABE LAUFZEIT A(I,2)
  - NEIN

WIEDERHOLE BIS N

- BERECHNEN UND RUNDEN DER ZINSEN A(I,3)
- BERECHNEN UND RUNDEN DES NEUEN KAPITALS A(I,4)

WIEDERHOLE BIS N

- FARBE UMSTELLEN
- AUSGABE ALLER WERTE
- WEITER DURCH TASTE "W"

FARBE ZURÜCKSTELLEN

ENDE

CODIERUNG:

```
5 SCNCLR
10 PRINT "EINFACHE ZINSRECHNUNG"
20 PRINT:PRINT
30 INPUT "WIEVIELE KAPITALIEN?";N
40 REM FELD ANLEGEN MIT N ZEILEN
50 REM UND 4 SPALTEN FUER
60 REM KAPITAL, LAUFZEIT, ZINS
70 REM UND NEUES KAPITAL
80 DIM A(N,4)
84 :
85 REM ES BEGINNT DIE EINGABE
86 :
90 FOR I=1 TO N
100 INPUT "ANFANGSKAPITAL  =";A(I,1)
105 IFA(I,1)<=0THEN PRINT"FALSCH":GOTO100
110 REM KAPITAL STEHT IN I.TER ZEILE
111 REM 1.TER SPALTE
120 INPUT "LAUFZEIT/TAGE   =";A(I,2)
130 REM LAUFZEIT STEHT IN I.TER ZEILE
131 REM 2.SPALTE
140 NEXT I
150 :
160 REM ES BEGINNT DIE BERECHNUNG
170 :
180 FOR I=1 TO N
200 REM ZINSEN SOLLEN NACH I.TER ZEILE
210 REM UND 3.TER SPALTE
220 A(I,3)=A(I,1)*A(I,2)*10/(360*100)
221 A(I,3)=INT(A(I,3)*100+0.5)/100
230 REM NEUES KAPITAL IN I.TER ZEILE
240 REM UND 4.TER SPALTE
250 A(I,4)=A(I,1)+A(I,3)
251 A(I,4)=INT(A(I,4)*100+0.5)/100
260 NEXT I
270 :
280 REM ES BEGINNT DIE AUSGABE
290 :
300 COLOR 1,3,5
310 FOR I=1 TO N
315 PRINT:PRINT
320 PRINT "ANFANGSKAPITAL =",A(I,1)
330 PRINT "LAUFZEIT       =",A(I,2)
340 PRINT "ZINSEN         =",A(I,3)
350 PRINT "NEUES KAPITAL  =",A(I,4)
355 PRINT:PRINT
360 PRINT "WEITER MIT TASTE 'W'"
370 GET P$
380 IF P$<>"W" GOTO 370
390 NEXT I
400 COLOR 1,1,0
410 END

READY.
```

8.7 Kontrollfragen

1. Wieviele Speicherplätze werden durch DIM Z (10) reserviert?

2. Beschreiben Sie eine eindimensionale Tabelle (= Vektor) und eine zweidimensionale Tabelle (= Matrix)!

3. Weshalb ist es sinnvoll, mit Tabellen die FOR ... TO ...Schleife zu benutzen?

4. Erläutern Sie die drei Komponenten im Zusammenhang zu dem Schlüsselwort COLOR!

5. Entwickeln Sie ein Programm, das es ermöglicht, bis zu 100 Anschriften getrennt nach Name, Straße, Postleitzahl und Wohnort einzugeben, zu speichern und wieder auszugeben!

Für Ihre Lösungen:

9. Fehlersuche und Fehlerbeseitigung

Lernziel: Der Leser soll grundsätzliche Wege der Fehlersuche und der Fehlerbeseitigung kennen und anwenden lernen. Dazu soll er mit verschiedenen zusätzlichen Funktionen des Gerätes vertraut gemacht werden.

BASIC-Schlüsselworte: HELP
TRACE
TRON
TROF
TRAP
RESUME

Bei der Programmierung muß man davon ausgehen, daß kein Programm „auf Anhieb" richtig läuft. Aus diesem Grund gehört zum Programmieren das Testen und die Fehlersuche, die oftmals mehr Zeit in Anspruch nehmen, als das Programmieren selbst.

Die Fehlerarten lassen sich in drei Gruppen unterscheiden:

- Syntax-Fehler

- Ablauf-Fehler

- Logische Fehler.

9.1 Syntax-Fehler

Das Wort Syntax kommt aus den Sprachwissenschaften; mit ihm werden diejenigen Regeln bezeichnet, die bei der Bildung von Sätzen aus Wörtern zu beachten sind. Gleiche Bedeutung hat das Wort Syntax auch in einer Programmiersprache.

Da die Programmiersprache vom Computer in die Maschinensprache übersetzt wird, muß er nach ganz genau festgelegten Regeln arbeiten: die Syntax.

Ist dieses Syntax nicht beachtet, so meldet sich der Computer mit der Fehlermeldung "SYNTAX-ERROR".

Der Computer unterbricht den Arbeitsablauf des Programms und verweigert die Übernahme des Befehls in den Befehlsspeicher, wenn noch Syntax-Fehler enthalten sind. Dem Programmierer bleibt nichts anderes übrig, als diese Syntax-Fehler über das Gerät, d.h. die Tastatur und den Monitor, zu beseitigen.

9.2 Ablauf-Fehler und logische Fehler

Wenn ein Programm vollständig, ohne Syntax-Fehler, im Programm-Speicher steht, so ist noch nicht sichergestellt, daß es auch ordnungsgemäß abläuft. Die Syntax-Prüfung kann nämlich nicht feststellen, ob die angegebenen Sprungbefehle richtig sind und die Sprungziele (als Zeilennummer) richtig angegeben wurden.

In diesem Falle meldet sich der Computer über den Monitor mit der Fehlermeldung

"UNDEF'D STATEMENT ERROR" oder auch "DIVISION BY ZERO ERROR",

wenn eine Variable keinen Inhalt zugewiesen bekommen hat.

Im einzelnen sind hierzu die entsprechenden Fehlermeldungen im Bedienungshandbuch nachzulesen; eine Kurzfassung befindet sich im Anhang **3.**

9.3 Zehn Tips zur Fehlerbeseitigung

9.3.1 Syntax-Fehler

1. Die Fehlermeldung eines SYNTAX-ERRORS wird angezeigt durch:

   "SYNTAX ERROR IN (Zeilennummer)".

   Der Programmierer kann sich nun über das Systemkommando LIST "Zeilennummer" diejenige Zeile aufrufen, in der der Syntax-Fehler ist. **Mit dem Cursor geht er auf diejenige Stelle, die falsch ist und übertippt den Fehler durch die richtige Darstellung.** Danach kann sofort die Taste RETURN gedrückt werden und die richtige, korrigierte Zeile wird in die Zentraleinheit aufgenommen. Es ist nicht notwendig, daß vor dem Drücken der Taste RETURN zum Zeilenende zurückgegangen wird.

2. Findet man den Syntax-Fehler nicht sofort, so bietet das System Commodore C116/C16/+4 eine besondere "HELP-Taste", die das Auffinden der fehlerhaften Stelle erleichtert. **Wenn die HELP-Taste gedrückt wird, dann wird die fehlerhafte Programmzeile aufgelistet, wobei der Teil mit dem Fehler blinkend dargestellt wird.**

   Durch richtiges Neu-Schreiben der gesamten Zeile oder durch Korrigieren über die Cursor-Steuertaste muß der Fehler beseitigt werden.

9.3.2 Logische Fehler

3. Nachdem die Syntax-Fehler beseitigt wurden, sollte man sich das Programm noch einmal ansehen, um festzustellen, ob sich in dem Programm Befehle befinden, die nicht hineingehören. **Dazu kann man sich mit dem Kommando-Wort "LIST" das Programm auf den Bildschirm auslisten.** Zweckmäßigerweise sollte man den Bildschirm vorher mit der Taste "CLEAR/HOME" löschen. Das Programm wird jetzt in schneller Folge auf dem Bildschirm dargestellt. Um den Ablauf auf dem Bildschirm zu verlangsamen, ist eine besondere Taste zu drücken; bei den Systemen C116/C16/+4 ist es die Taste mit dem „Commodore-Zeichen".

   Da der Bildschirm nur eine begrenzte Anzahl von Zeilen anzeigen kann, muß man sich bei größeren Programmen die Programme abschnittsweise ansehen.

4. Um das Gesamtprogramm vor Augen zu haben, und nicht nur einen Teil, wie auf dem Monitor möglich, sollte man sich bei größeren Programmen **das gesamte Programm auf dem Drucker auslisten lassen.** Dies kann durch folgende Befehle geschehen (siehe auch 10.6):

   OPEN 1,4:CMD1:LIST [RETURN]

   Nach dem Auslisten des Programms ist einzugeben:

   PRINT # 1 [RETURN]

   CLOSE 1 [RETURN]

   Das ausgedruckte Blatt kann jetzt in Ruhe auf Fehler untersucht werden.

5. **Sind nun alle offensichtlichen Fehler korrigiert, und das Programm läuft noch immer nicht, so sollte man versuchen, zunächst Teile des Programms zum „Laufen" zu bringen.**

   Hierzu kann man an den jeweils sinnvollen Stellen im Programm das BASIC-Schlüsselwort "STOP" einfügen. Das Programm hält dann an dieser Stelle an, ohne daß Daten oder Variable zerstört werden.

   Die Inhalte der Variablen können jetzt abgefragt werden, und zwar über das BASIC-Schlüsselwort "PRINT", das jetzt aber im Direktbetrieb, d.h. als Systemkommando, benutzt wird. Dazu ist folgendes einzugeben:

   PRINT (Variablenname)

   Mit diesem Befehl wird der Inhalt der genannten Variablen angegeben und man kann prüfen, ob er logisch erscheint. Das Programm kann jederzeit mit dem Systemkommando "CONT" fortgesetzt werden.

   Bei den Commodore-Geräten kann das BASIC-Schlüsselwort "PRINT" durch ein Sonderzeichen ersetzt werden, und zwar durch das "?". Damit kann die Abfrage der Variablen wie folgt lauten:

   ? (Variablenname)

   Sinnvolle Stellen, an denen der STOP-Befehl stehen soll, sind beispielsweise: vor einem Verzweigungsbefehl und nicht dahinter.

6. **Bei großen Programmen sollte man dem Einfügen von "STOP"-Befehlen eine besondere Logik zukommen lassen,** d.h. man sollte STOP-Befehle beispielsweise immer mit einer bestimmten Zeilennummer versehen, also beispielsweise immer mit einer 9 am Ende.

   Vermieden werden sollte, diese STOP-Befehle durch einen Doppelpunkt an das Zeilenende eines Abschnittes zu setzen. Der Vorteil einer eigenen Zeilennummer besteht darin, daß diese Programmzeilen viel einfacher zu löschen sind, wenn das Programm läuft.

7. **Bei der Analyse der Variablen, die in dem betreffendem Programm enthalten sind, kann es zweckmäßig sein, durch zusätzliche PRINT-Befehle im Programm selbst nicht nur die Variablen, sondern auch den Variablennamen ausdrucken zu lassen.** In dem Prüfbefehl sollte man dann zusätzlich das jeweilige Wort in Anführungsstrichen hinter dem BASIC-Schlüsselwort PRINT setzen. Ein solcher Befehl könnte wie folgt aussehen:

   Zeilennummer PRINT "Wert";N

   Es kann zweckmäßig sein, im Programmablauf von außen über die Tastatur mit der Taste "RUN/STOP" abzustoppen und den Inhalt der Variablen zu prüfen.

8. Hat man einzelne Programmteile geprüft, so startet man im allgemeinen das Programm mit dem Systemkommando "RUN". **Es kann von Vorteil sein, daß man anstelle dieses Systemkommandos das Systemkommando "GOTO Zeilennummer" einfügt.** Dadurch werden vorher geprüfte Teile nicht nochmals durchlaufen und nur derjenige Programmteil, der Schwierigkeiten bereitet, geprüft und kann korrigiert werden.

9. **Kommt man im Zuge der Analyse eines Programms zu dem Ergebnis, daß eventuell eine bestimmte Formel nicht richtig abgearbeitet wird bzw. nicht richtig programmiert wurde, so sollte man den Variablen der Formel einfache Zahlenwerte zuweisen.** Das Ergebnis kann man oftmals bereits im Kopf vorweg nehmen und sehen, ob es sich tatsächlich einstellt.

   Hat man einfache Zahlenwerte geprüft und die Formel arbeitet richtig, so ist sie auf Extremwerte zu prüfen, d.h. es ist zu prüfen, ob eventuell negative Werte, Nullwerte u.a., abgewiesen werden oder ob diese Extremwerte zu Fehlern führen.

10. Das System Commodore C116/C16/+4 bietet, neben den allgemeinen Möglichkeiten der Fehlerbeseitigung noch einige besondere Möglichkeiten und BASIC-Schlüsselworte, um Fehlern auf die Spur zu kommen. Dies ist der "TRACE"- und der "TRAP"-Befehl. Beide dienen dazu, um logische Fehler schneller auffinden zu können.

    Der TRACE-Befehl ist in den Systemen C116/C16/+4 in besondere BASIC-Befehlsworte gefaßt worden.

Dies ist das BASIC-Schlüsselwort "TRON" (= TRACE ON), das am Anfang derjenigen Schleife bzw. desjenigen Teiles des Programmes steht, der geprüft werden soll. Am Ende dieses Teiles des Programmes ist das BASIC-Schlüsselwort "TROFF" (= TRACE OFF) anzubringen.

Diese beiden Schlüsselworte bewirken, daß alle Befehle, die zwischen ihnen stehen, abgearbeitet werden und jeweils, vor der Ausgabe eines Ergebnisses, angezeigt wird, wie die Reihenfolge der Zeilen, die abgearbeitet wurden, war.

Eine weitere Besonderheit der C116/C16/+4-Systeme ist, daß sie das BASIC-Schlüsselwort "TRAP" in Verbindung mit dem BASIC-Schlüsselwort "RESUME" aufgenommen haben.

In Verbindung mit diesen Worten wurden bestimmen Variablenbezeichnungen besondere Werte und Fehlermeldungen zugewiesen:

EL = „Zeilennummer, in der der Fehler auftrat"

ER = „die eigentliche Fehlermeldung in Nummernschreibweise"

ERR (Fehlernummer)
= „die Decodierung der Fehlermeldung aus dem Dezimalsystem in Buchstabenschreibweise".

Mit dem Systemkommando PRINT ERR (Zeilennummer) kann auch im Direktbetrieb der Fehler abgefragt werden.

Wird das Befehlswort TRAP in ein BASIC-Programm eingefügt, so muß unbedingt die Zeilennummer hinter dem BASIC-Schlüsselwort TRAP stehen. Taucht ein Fehler auf, so wird in die, hinter dem TRAP stehende Zeile verzweigt. Dort sollte eine kleine Fehlerroutine stehen.

Wurde über die TRAP-Anweisung ein Fehler analysiert und beseitigt, so wird mit dem BASIC-Schlüsselwort "RESUME" das Programm mit demjenigen Befehl fortgesetzt, welcher den Fehler verursachte.

Mit dem BASIC-Schlüsselwort "RESUME NEXT" wird das Programm mit derjenigen Zeile fortgesetzt, die hinter der Zeile steht, die den Fehler erzeugte. Hinter dem BASIC-Schlüsselwort "RESUME" kann aber auch eine Zeilennummer stehen. Dann funktioniert dieses Schlüsselwort wie eine GOTO-Anweisung und veranlaßt im Fehlerfall die Programmfortführung ab der angegebenen Zeile.

Damit läuft eine Fehlerbehandlung mit dem BASIC-Schlüsselwort TRAP in vier Schritten ab:

1. Fehlerbehandlung eröffnen

2. Fehlerroutine aufstellen
   Fehlercode in ER abfragen
   Fehlerhinweise ausgeben mit EL und der Meldung ERR

3. Programmablauf fortsetzen mit RESUME

4. Fehlerbehandlung abschließen.

9.4 Kontrollfragen

1. Welche Fehlergruppen werden unterschieden?

2. Welche Wirkung hat die Taste HELP?

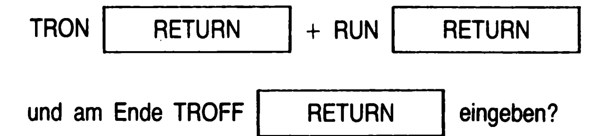
3. Überlegen Sie sich, wie man Zahlen sortieren und in aufsteigender Reihe darstellen kann!

4. Was bedeutet „Platztausch"?

5. Versuchen Sie ein Sortierprogramm zu schreiben.

6. Wie muß das Sortierprogramm geändert werden, damit auch Buchstaben bzw. Namen sortiert werden können?

7. Welche Wirkung stellt sich ein, wenn Sie



Für Ihre Lösungen:

10. Die vollständige Bearbeitung eines Programms

Lernziel: Der Leser soll erkennen und lernen, wie ein Programm in BASIC erstellt wird; wie es über die Tastatur in die Zentraleinheit eingegeben wird; und wie die zugehörigen Peripheriegeräte das Programm speichern und zur erneuten Abarbeitung zur Verfügung stellen können.

BASIC-Schlüsselwort: alle bisher besprochenen!

BASIC-Systemkommandos: LIST
SAVE
LOAD

Struktursymbol: alle bisher besprochenen!

10.1 Das Programm „ZINS"

Das bisherige BASIC-Programm, das wir durchgehend in jedem Kapitel programmiert haben und das jeweils dasselbe Ergebnis zum Ziel hatte, soll jetzt nochmals erweitert werden. Es sei betont, daß jede der früheren Lösungen für sich gesehen korrekt ist, aber jeweils einer anderen Aufgabenstellung genügt. Diese Aufgabenstellung ist aber in der Praxis meist unterschiedlich, so daß man ein Programm so schreiben sollte, daß es fast allen Situationen genügt.

10.1.1 Problem- und Lösungssuche

1. Problemstellung:
   Das folgende Programm soll die Möglichkeit bieten, einfache Bankzinsen per Formel zu berechnen; es soll aber gleichzeitig die anderen, in der Formel enthaltenen Größen, d.h. die Tage/Laufzeit, den Zinssatz und das Anfangskapital, bestimmen können.

2. Problemanalyse:
   Wir müssen die Formel der einfachen Bankzinsen benutzen; die vorhandenen Werte eingeben und die gewünschte Größe bestimmen können.

3. Der Lösungsweg:
   Die Bankzinsformel ist einem Lehrbuch zu entnehmen und nach den jeweiligen Größen umzustellen:

   Z = KØ*P*T/(1ØØ*36Ø)
   KØ = P*T/(Z*1ØØ*36Ø)
   P = KØ*T/(Z*1ØØ*36Ø)
   T = KØ*P/(Z*1ØØ*36Ø)

   Wir benutzen folgende Variable:

   Z = Zinsen
   KØ = Anfangskapital
   P = Zinssatz
   T = Laufzeit in Tagen
   KN = Neues Kapital

   Das neue Kapital wird durch Addition der Zinsen zum Anfangskapital berechnet.

4. Der Lösungsalgorithmus:
   Der Benutzer sucht aus, welche Werte er berechnen möchte und gibt die vorhandenen Werte ein.

10.1.2 Das Struktogramm

BILDSCHIRM FREI / ÜBERSCHRIFT

WIEDERHOLE BIS ENDE GEWÜNSCHT

AUSWAHLNUMMER EINGEBEN

1 = ZINSEN	2 = ANFANGSKAPITAL	3 = ZINSSATZ	4 = LAUFZEIT	5
BILDSCHIRM FREI	BILDSCHIRM FREI	BILDSCHIRM FREI	BILDSCHIRM FREI	
WIEDERHOLE BIS RICHTIG EINGABE: ANF. KAP. = KØ ZINSSATZ = P LAUFZEIT = T	WIEDERHOLE BIS RICHTIG EINGABE: ZINSEN = Z ZINSSATZ = P LAUFZEIT = T	WIEDERHOLE BIS RICHTIG EINGABE: ZINSEN = Z ANF. KAP. = KØ LAUFZEIT = T	WIEDERHOLE BIS RICHTIG EINGABE: ZINSEN = Z ANF. KAP. = KØ ZINSSATZ = P	
BERECHNE UND RUNDE: NEUES KAPITAL	BERECHNE UND RUNDE: ANFANGSKAPITAL	BERECHNE UND RUNDE: ZINSSATZ	BERECHNE UND RUNDE: LAUFZEIT	
AUSGABE: ZINSEN, NEUES KAPITAL	AUSGABE: ANFANGSKAPITAL	AUSGABE: ZINSSATZ	AUSGABE: LAUFZEIT	
				ENDE

```
5 SCNCLR
10 PRINT "EINFACHE BANKZINSEN"
20 PRINT "U.A. BERECHNUNGEN"
30 PRINT:PRINT
40 PRINT "BITTE WAEHLEN SIE DIE"
41 PRINT "GEWUENSCHTE BERECHNUNG AUS!"
42 PRINT
50 PRINT "ZINSEN            = 1"
51 PRINT
60 PRINT "ANFANGSKAPITAL    = 2"
61 PRINT
70 PRINT "ZINSSATZ          = 3"
71 PRINT
80 PRINT "LAUFZEIT IN TAGEN= 4"
81 PRINT
90 PRINT "ENDE              = 5"
91 PRINT
100 INPUT A
110 ON A GOTO 200,300,400,500,120
120 END
200 SCNCLR:PRINT:INPUT "ANFANGSKAPITAL =";K0
210 INPUT "ZINSSATZ       =";P
220 INPUT "LAUFZEIT/TAGE  =";T
225 IF K0<=0 OR P<=0 OR T<=0 THEN PRINT"FALSCH!":GOTO 205
230 Z=K0*P*T/(100*360)
231 Z=INT(Z*100+0.5)/100
240 KN=K0+Z
241 KN=INT(KN*100+0.5)/100
250 PRINT:PRINT
260 PRINT "NEUES KAPITAL  =";KN;"DM"
270 PRINT "ZINSEN         =";Z;"DM"
280 PRINT:PRINT
290 GOSUB 600
300 SCNCLR:PRINT:INPUT "ZINSEN          =";Z
310 INPUT "ZINSSATZ       =";P
320 INPUT "LAUFZEIT/TAGE  =";T
325 IF  Z<=0 OR P<=0 OR T<=0 THEN PRINT"FALSCH!":GOTO 305
330 K0=Z*100*360/(P*T)
331 K0=INT(K0*100+0.5)/100
350 PRINT:PRINT
360 PRINT "ANFANGSKAPITAL  =";K0;"DM"
380 PRINT:PRINT
390 GOSUB 600
400 SCNCLR:PRINT:INPUT "ZINSEN          =";Z
410 INPUT "ANFANGSKAPITAL =";K0
420 INPUT "LAUFZEIT/TAGE  =";T
425 IF  Z<=0 OR K0<=0 OR T<=0 THEN PRINT"FALSCH!":GOTO 405
430 P=Z*100*360/(K0*T)
431 P=INT(P*100+0.5)/100
450 PRINT:PRINT
460 PRINT "ZINSSATZ        =";P;"%"
480 PRINT:PRINT
490 GOSUB 600
500 SCNCLR:PRINT:INPUT "ZINSEN          =";Z
510 INPUT "ANFANGSKAPITAL =";K0
520 INPUT "PROZENTSATZ    =";P
525 IF  Z<=0 OR K0<=0 OR P<=0 THEN PRINT"FALSCH!":GOTO 505
530 T=Z*100*360/(K0*P)
531 T=INT(T*100+0.5)/100
550 PRINT:PRINT
560 PRINT "LAUFZEIT        =";T;"TAGE"
580 PRINT:PRINT
600 INPUT "NOCH EINE BERECHNUNG?J/N";B$
605 SCNCLR
610 IF B$="J" THEN 30
620 IF B$="N" THEN 120

READY.
```

10.2 Die Arbeit am Gerät

10.2.1 Gerät bereitstellen

Unsere Gerätekonfiguration ist entsprechend dem Bedienungshandbuch anzuschließen. Nach Einschalten des Monitors und der Zentraleinheit erscheint auf dem Bildschirm die folgende Meldung

```
COMMODORE BASIC V3.5 12277 BYTES FREE

READY
C
```

Der Wert 12.277 Bytes free (für den C116 und C16) gibt an, daß im Arbeitsspeicher (RAM = RANDOM ACCESS Memory) 12.277 Plätze für unser Programm zur Verfügung stehen.

Wurde bereits zuvor mit dem Commodore gearbeitet und befinden sich noch ein Programm oder Daten im Hauptspeicher, so müssen wir vorher diese Informationen löschen. Dies geschieht durch das Systemkommando NEW. Über die Tastatur tippen wir das Wort NEW als Einzelbuchstaben und drükken anschließend die Taste RETURN. **Auf dem Bildschirm erscheint dann die Meldung READY und unter dem R steht der blinkende Cursor, der zum Ausdruck bringt, daß wir jetzt arbeiten können.**

10.2.2 Das Eintippen des Programms

Der Arbeitsspeicher des Commodore ist jetzt leer und bereit, unser Programm aufzunehmen. **Wir tippen das Programm Zeile für Zeile ein, wobei wir nach jeder Zeile unbedingt die Taste RETURN drücken müssen,** denn damit wird die Zeile, die auf dem Bildschirm erscheint, in die Zentraleinheit „abgeschickt".

Nachdem wir das Programm vollständig eingetippt haben, tippen wir das Schlüsselwort LIST ein und drücken anschließend die Taste RETURN. Damit erscheint das gesamte von uns eingegebene Programm vollständig auf dem Bildschirm.

10.2.3 Die Ausführung des Programms

Über die Tastatur tippen wir das Wort RUN ein und drücken anschließend die RETURN-Taste.

HINWEIS: Verwechseln Sie nicht das Wort RUN mit der Taste RUN/STOP, die eine andere Wirkung hat.

Das Programm wird jetzt so ausgeführt, wie es dem Commodore durch die Anweisungsnummern eingegeben wurde. Bei der Abarbeitung des Programms wird mit der niedrigsten Zeilennummer begonnen.

10.3 Die Arbeit mit Peripheriegeräten

10.3.1 Speichern und Laden von Programmen über Datasette

Wenn Sie eine **Datasette zu Ihrem Commodore** gekauft haben, so sollten Sie jetzt eine Kassette zur Hand nehmen. Dabei können Sie entweder spezielle Computer-Kassetten oder aber auch eine beliebige Musik-Kassette benutzen. Bitte beachten Sie, daß der Computer nicht erkennt, ob er auf dem Magnetteil des Bandes ist oder nicht. Aus diesem Grund sollten Sie das Band bis zur Markierung des Magnetteiles von Hand vorspulen. Der Computer prüft nur, ob die Tasten gedrückt sind.

Wir geben unserem Programm den Namen „Zins". Um das Programm nun speichern zu können, geben wir über die Tastatur eine Befehlsfolge:

SAVE "ZINS" [RETURN]

Der Computer meldet sich über den Monitor mit der Befehlsfolge:

PRESS PLAY AND RECORD ON TAPE.

Jetzt sind gleichzeitig die beiden Tasten Rekord und Play auf dem Gerät zu drücken. Über den Monitor erscheint die Meldung

OK SAVING ZINS

Ist der Vorgang beendet, so erscheint:

READY

und darunter steht der blinkende Cursor.

Damit ist Ihr Programm gespeichert und Sie können es jederzeit wieder abrufen. Das Abrufen erfolgt in einen freien Arbeitsspeicher. Es können nämlich niemals zwei Programme gleichzeitig im Arbeitsspeicher stehen. Wir geben also das Wort NEW ein und es erscheint auf dem Bildschirm die Nachricht READY. Damit befindet sich kein Programm mehr im Arbeitsspeicher und wir können unser Zinsprogramm von der Kassette, nachdem wir sie zurückgespult haben, laden.

Dies geschieht durch das Systemkommando LOAD "ZINS" und danach drücken der Taste RETURN. Über den Monitor erscheint die Meldung PRESS PLAY ON TAPE und nachdem wir die Taste Play gedrückt haben, erscheint OK SEARCHING FOR ZINS. Danach erscheint die Meldung FOUND ZINS, LOADING und das Wort READY, wenn das Programm geladen ist. Unter dem Wort READY ist wiederum der blinkende Cursor, der angibt, daß das Programm zur Verfügung steht.

Die Befehlsfolge lautet:

LOAD "ZINS" [RETURN]

Über das Systemkommando LIST können wir uns das Programm auf dem Bildschirm auslisten lassen.

Wir können ein Programm, das wir auf Kassette gespeichert haben, durch das Systemkommando VERIFY überprüfen lassen, und zwar darauf hin, ob es richtig von der Zentraleinheit auf das Band übertragen wurde. Wir geben also das Wort VERIFY ein und drücken anschließend die Taste RETURN. Auf dem Bildschirm erscheint die Meldung "PRESS PLAY ON TAPE" und nach dem Drükken der Taste RETURN das Wort OK SEARCHING.

Die Befehlsfolge lautet:

VERIFY "ZINS" [RETURN]

Danach erscheint die Meldung FOUND ZINS, darunter VERIFYING und anschließend die Nachricht OK. Sollte hier ein Fehler auftreten, so erscheint die Fehlermeldung "VERIFY ERROR" und wir versuchen den SAVE-Vorgang erneut. Bei einem nochmaligen Fehler sollte man eine andere Kassette verwenden.

Eine Besonderheit der Commodore-Geräte besteht darin, daß wir den Namen des Programms nicht unbedingt angeben müssen. Es wird dann dasjenige Programm gelesen, das als nächstes auf dem Band gefunden wird.

10.3.2 Speichern und Laden von Programmen auf Diskette

Die Datasette ist zwar ein billiges aber doch recht langsames Peripheriegerät zum Speichern von Programmen. **Besser geeignet, weil schneller, ist die Diskette als Speichermedium und mit ihr die Flopydisk als Hardware, die die Diskette verwaltet.**

Wenn Sie eine Diskette im Handel kaufen, so ist diese Diskette für eine Vielzahl von unterschiedlichen Systemen geeignet. Für das System Commodore ist eine besondere Aufzeichnung notwendig, da das Diskettenlaufwerk, z.B. Flopy-"VC1540 oder VC1541", eine andere Aufzeichnung auf der Diskette hat als die anderen Systeme. Die Diskette als allgemeine Hardware soll aber für alle Systeme geeignet sein und muß deshalb für das jeweilige System aufbereitet werden.

Dies geschieht bei Commodore durch das Systemkommando HEADER, das eine Diskette formatiert.

Dabei ist folgende Befehlsfolge (beispielsweise) einzugeben:

HEADER "COMMODORE",DØ,IØ1

und drücken der Taste RETURN. Unter dieser Zeile erscheint auf dem Bildschirm ARE YOU SURE? Sie haben jetzt die Möglichkeit, ja oder nein zu sagen. Bitte beachten Sie, es ist das englische YES oder NO zu benutzen. Diese Frage ist deswegen vorgeschaltet, weil über das Wort HEADER alte Disketten, die nicht mehr benötigt werden, für neue Zwecke zur Verfügung gestellt werden können. Wenn Sie "Y" eingeben und die Taste RETURN drücken, so ist der blinkende Cursor auf dem Bildschirm für ungefähr 1 1/4 Minuten verschwunden, da das Gerät die zur Verfügung gestellte Diskette formatiert und dabei 35 x kreisrunde Spuren auf die Diskette schreibt, die später mit den zugehörigen Suchbefehlen wieder gefunden werden können. Die Verwaltung einer solchen Diskette geschieht im Betriebssystem des Computers. Erscheint der blinkende Cursor wieder auf dem Bildschirm, so können wir prüfen, ob die Diskette uns zur Verfügung steht, indem wir eingeben:

DIRECTORY [RETURN]

Auf dem Bildschirm erscheint die Meldung 0"COMMODORE"012A und darunter 664 BLOCKS FREE und darunter das Wort READY.

Jetzt können wir das Programm auf der Diskette speichern, und zwar mit dem Befehlswort DSAVE und dem zugehörigen Programmnamen. Nach dem Drücken der Taste RETURN wird das Programm auf die Diskette geschrieben und nach Beendigung des Schreibvorganges erscheint wiederum das Wort READY auf dem Monitor.

Die Befehlsfolge lautet:

DSAVE "ZINS" | RETURN |

Geben wir jetzt wiederum das Befehlswort DIRECTORY ein, so erscheint auf dem Bildschirm folgende meldung:

O"COMMODORE"012A

2"ZINS"PRG

662BLOCKS FREE

READY.

Damit ist das Programm gespeichert.

Mit dem BASIC-Schlüsselwort DLOAD und dem Programmnamen und dem Drücken der Taste RETURN, können wir das Programm wiederum in die Zentraleinheit einlesen.

Wir geben zuvor das Schlüsselwort NEW ein und leeren damit den Arbeitsspeicher. Danach geben wir als Befehlsfolge ein:

DLOAD"ZINS" | RETURN |

Es erscheint die Meldung

SEARCHING FOR 0:ZINS

LOADING

READY.

Damit ist das Programm in die Zentraleinheit geladen.

Einen solchen Ladevorgang können mit wir der Taste RUN/STOP abbrechen.

Programme und Daten, die auf Datasette abgespeichert wurden, können durch besondere Befehle gelöscht werden. Im einzelnen ist hierbei das Bedienungshandbuch der Flopydisk zu Rate zu ziehen.

TASTEN	MONITOR	WIRKUNG
F2 "Programm-Name" RETURN	DLOAD "Programm-Name"	Lädt das genannte Programm in die ZE
SHIFT + F5 "Programm-Name" RETURN	DSAVE "Programm-Name"	Das genannte Programm wird auf der Diskette gespeichert
F3 RETURN	"Disketteninhalt"	Listet das Inhaltsverzeichnis der Diskette
SHIFT+RUN/STOP	DL"*	Lädt das erste Programm auf der Diskette in die ZE

Abb. 21: Übersicht über die Kurzeingabe von Diskettenbefehlen

10.3.3 Auflisten von Programmen über den Drucker

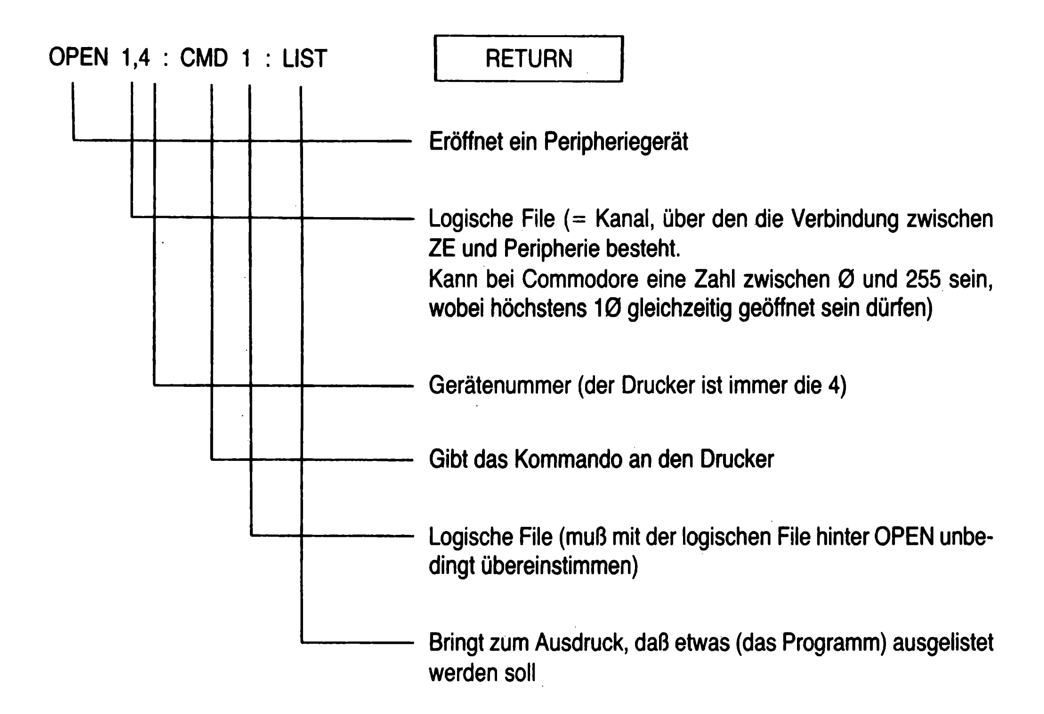
Will man mit dem Drucker des Commodore arbeiten, so muß man ihn dem System zur Verfügung stellen. Dies bedeutet nicht nur, daß er als Hardware zur Verfügung steht, sondern man muß einen Kanal (eine Datenübertragungslinie) haben, um die Daten, d.h. in unserem Falle das Programm von der Zentraleinheit auf den Drucker übertragen zu können. Bei der Datasette und der Flopydisk hat dies das Gerät selbst erledigt. **Bei dem Drucker müssen wir dafür sorgen, daß er dem Gerät zur Verfügung steht.** Dies geschieht durch das Systemkommando OPEN; dieses Systemkommando ist zugleich ein BASIC-Schlüsselwort. Nachdem die Druckarbeit abgeschlossen ist, muß die Verbindungslinie (der Kanal) zum Drucker geschlossen werden; dies geschieht durch das Systemkommando und BASIC-Schlüsselwort CLOSE.

Mit dem Systemkommando CMD wird das Kommando an den Drucker abgegeben, der dann das Schreiben des Programmes selbst verwaltet.

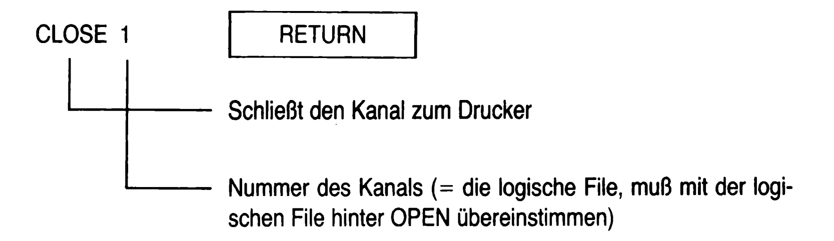
Es ist noch anzugeben, was der Drucker zu tun hat; ist das Programm aufzulisten, so geschieht das über das Systemkommando LIST.

Insgesamt lautet die Befehlsfolge:

Schreiben eines Programmes auf dem Drucker



Der LIST-Vorgang wird auf dem Drucker durch das Wort READY als beendet erklärt. Auf dem Monitor ist das Wort READY nicht zu sehen, sondern lediglich der blinkende Cursor; d.h. wir müssen uns die ZE wieder zur Verfügung stellen. Dies geschieht durch eine CLOSE-Anweisung wie folgt:



10.4 Kontrollfragen

1. Erläutern Sie die gesamte Befehlsfolge zum Sichern, Überprüfen und Laden eines Programms über Kassette!
2. Welche Bedeutung hat das Wort HEADER für die Disketten?
3. Welches sind die wesentlichen Diskettenbefehle?
4. Erläutern Sie die wichtigsten Druckerbefehle!

Für Ihre Lösungen:

ANHANG 1:

Lösung zu den Kontrollfragen Kapitel 1:

1. Lösung siehe Seite 12.

2. Lösung siehe Seite 12, 14.

3. Ein Computer kann nur diejenigen Werte, die eingegeben wurden, bearbeiten und das Ergebnis der Bearbeitung ausgeben.

4. Hierzu gibt es keine eindeutige Lösung; alle Punkte sind wichtig und sollten unbedingt bedacht werden.

5. Lösung siehe Seite 19, 20.

Lösungen zu Kapitel 2:

1. Die Meldung bedeutet, daß die BASIC-Version 3.5 in diesem Gerät verwirklicht wurde, d.h. die allgemein normierte Programmiersprache BASIC hat hier eine leichte Veränderung gefunden.

2. Lösung siehe Seite 23.

3. Lösung siehe Seite 24.

4. Der wesentliche Unterschied zwischen einem Systemkommando und einem BASIC-Befehl besteht darin, daß ein Systemkommando niemals eine Anweisungsnummer hat. Ein BASIC-Befehl dagegen muß stets eine Anweisungsnummer haben, da entsprechend der Reihenfolge der Anweisungsnummern das BASIC-Programm abgearbeitet wird.

5. Der Cursor gibt diejenige Stelle auf dem Bildschirm an, auf der zum jeweiligen Zeitpunkt gearbeitet werden kann.

6. Lösung siehe Seite 26 f.

7. Lösung siehe Seite 30.

8. Lösung siehe Seite 29.

9. Lösung siehe Seite 29.

Lösung der Kontrollfragen zu Kapitel 3:

1. Eine Variable ist ein Platzhalter, d.h. ein Name für einen Wert, der im Verlauf des Abarbeitens eines Programms zugewiesen wird; eine Konstante ist ein Wert in einem Feld, das stets den gleichen Inhalt hat, der im Verlauf des Programms benutzt wird.

2. Lösung siehe Seite 35.

3. Lösung siehe Seite 36.
   Integer Variable werden entweder mit der Ankürzung INT oder mit einem %-Zeichen versehen; Dezimalzahlvariable haben keine besonderen Sonderzeichen; Stringvariable werden mit dem Sonderzeichen S versehen.

4.

```
10 INPUT "1.ZAHL";A
20 INPUT "2.ZAHL";B
30 PRINT
40 PRINT "DIE ERSTE  ZAHL WAR";A
50 PRINT "DIE ZWEITE ZAHL WAR!;B


READY.
```

5. Lösung siehe Seite 41.

6.
```
10 INPUT "1.ZAHL";A
20 INPUT "2.ZAHL";B
30 C=A+B
40 PRINT "DAS ERGEBNIS =";C

READY.
```

7. Lösung siehe Seite 46.

8. Die RETURN-Taste muß nach jeder Zeile gedrückt werden, damit der Inhalt der Zeile, die bisher nur auf dem Bildschirm steht, von dem Bildschirm in die Zentraleinheit übernommen wird.

Lösung zu den Kontrollfragen des Kapitels 4:

1. Lösung siehe Seite 50.

2. Eine Programmschleife ist ein Teil eines Programmes, der mehrfach abgearbeitet werden soll, d.h. es ist beispielsweise eine Formel, die mit unterschiedlichen Werten immer neu berechnet werden soll. Man könnte diese Formel jeweils neu programmieren, was bedeuten würde, daß der Arbeitsaufwand recht hoch ist. In einer Programmschleife wird eine solche Formel nur ein einziges Mal programmiert und dann über die Schleifensteuerung in der entsprechenden Anzahl abgearbeitet.

3. Lösung siehe Seite 52.

4. Lösung siehe Seite 52 ff.

5.
```
10 PRINT "ADDITION DER ZAHLEN VON 1 BIS 10"
20 PRINT
30 FOR I=1 TO 10
40 S =S+I
50 PRINT "ZWISCHENERGEBNIS =";S
60 NEXT I
70 PRINT "GESAMTERGEBNIS =";S

READY.
```

6. Der Schleifenzähler ist eine Variable, die jeweils um eins (andere Möglichkeiten gibt es über das BASIC-Schlüsselwort STEP) erhöht wird. Da der Schleifenzähler eine Variable ist, kann der Inhalt dieser Variablen unter ihrem Namen (in unserem Beispiel I) abgerufen und auch für Berechnungen benutzt werden.

7.

```
10 PRINT "ADDITION DER ZAHLEN VON 1 BIS N"
20 PRINT
25 INPUT"WIEVIELE ZAHLEN SOLLEN ADDIERT WERDEN?";N
30 FOR I=1 TO N
40 S =S+I
50 PRINT "ZWISCHENERGEBNIS =";S
60 NEXT I
70 PRINT "GESAMTERGEBNIS =";S

READY.
```

8. Der Cursor läßt sich innerhalb von Anführungsstrichen über die Taste DEL/INS in Verbindung mit der Taste SHIFT steuern.

Lösungen zu den Kontrollfragen des Kapitels 5:

1. Lösung siehe Seite 60.

2. Vergleichsoperatoren sind Symbole, die zwei Variable oder auch Konstante miteinander vergleichen. Ihre Bedeutung liegt darin, daß an den Vergleich eine Verzweigung nach einer beliebigen Programmzeile im BASIC-Programm angeschlossen werden kann.

3. Lösung siehe Seite 63 ff.

4. Wir sollen auf jedes Feld des Schachbretts immer die doppelte Anzahl von Körnern legen, wie auf dem vorhergegangenen Feld. Das bedeutet, daß wir die vorhergegangene Zahl von Körnern mit 2 multiplizieren müssen. Das Problem läßt sich mit einer IF...THEN...-Schleife lösen oder über eine FOR...TO...-Schleife. Zu beachten ist jedoch bei beiden, daß wir bei dem ersten Feld ein Korn dazulegen müssen, da der Computer jedes Feld von Anfang an gleich 0 setzt. Dies würde bedeuten, daß wir in Zeile 80 bzw. Zeile 160 des folgenden Programms jeweils die Multiplikation mit 0 und damit auch das Ergebnis 0 hätten. Deshalb sind in unserem Beispiel die Zeilen 60 und 140 unbedingt erforderlich.
   (Codierung S. 134)

```
10 PRINT "ANZAHL VON GETREIDEKOERNERN AUF DEM SCHACHBRETT"
20 PRINT:PRINT
25 :
30 REM ERSTE LOESUNG MIT IF ... THEN
40 REM VARIABLE: K=KOERNER SK=SUMME DER KOERNER Z=ZAEHLER
50 REM AM ANFANG LIEGT AUF DEM 1. FELD EIN KORN
60 K=1:SK=1
70 Z=Z+1
80 K=K*2:SK=SK+K
90 IF Z=63 THEN 110
100 GOTO 70
110 PRINT "1.LOESUNG: ES SIND";SK;"KOERNER"
120 :
130 REM ZWEITE LOESUNG MIT FOR ... TO
140 K=1:SK=1
150 FOR I=1 TO 63
160 K=K*2:SK=SK+K
170 NEXT I
180 PRINT "2. LOESUNG: ES SIND";SK;"KOERNER"
190 END


READY.
```

5.

```
5 SCNCLR:PRINT
10 PRINT "ZAHLENRATEN"
20 PRINT:PRINT
30 PRINT "EIN SPIELER GIBT EINE ZAHL EIN"
40 PRINT "DER ANDERE MUSS SIE ERRATEN!"
50 PRINT "ZUR ERLEICHTERUNG SOLL DER "
60 PRINT "ZAHLENBEREICH ANGEGEBEN WERDEN"
70 PRINT "GEWONNEN HAT WER DIE WENIGSTEN RATEVERSUCHE HAT":PRINT
80 PRINT "■SPIELER 1 BITTE UMDREHEN■"
90 PRINT "■SPIELER 2 ZAHL EINGEBEN ■"
100 PRINT:PRINT
110 PRINT:PRINT "ES GEHT LOS!!!"
120 PRINT:PRINT
130 INPUT "RATEZAHL????";A
140 SCNCLR
150 PRINT:INPUT "WELCHE ZAHL WIRD GERATEN?";B
160 REM WIR ZAEHLEN DIE VERSUCHE
161 Z=Z+1
170 IF A=B THEN 200
180 IF A<B THEN PRINT "DIE ZAHL WAR ZU GROSS!!!":GOTO 150
190 IF A>B THEN PRINT "DIE ZAHL WAR ZU KLEIN!!!":GOTO 150
200 PRINT "RICHTIG!":PRINT
210 PRINT "ES WAREN";Z;"VERSUCHE!"
220 INPUT "NOCH EIN SPIEL?J/N";C$
230 IF C$="J" THEN Z=0:GOTO 110
240 IF C$="N" OR C$<>"J" THEN END


READY.
```

Lösungen zu Kapitel 6:

1. In einer Warteschleife kann das Programm angehalten werden, ohne daß es in seinem Ablauf unterbrochen wird. Eine Warteschleife wird dann programmiert, wenn beispielsweise auf einem Peripheriegerät von Hand Operationen durchgeführt werden müssen, wie das Einlegen einer anderen Diskette oder das richtige Einspannen von Papier auf dem Drucker.

2. Mit dem Schlüsselwort INPUT können Daten in das Programm eingegeben werden, die anschließend in Formel bearbeitet werden. Eine weitere Möglichkeit, um Daten in das Programm einzugeben, ist das BASIC-Schlüsselwort GET; hier wird ein Zeichen von der Tastatur abgerufen.

3. Das BASIC-Schlüsselwort DATA muß mit einer Anweisungsnummer versehen werden und im Programm stehen. Alle Werte, die hinter dem Wort DATA stehen, sind Informationen, die in Variable übernommen werden können. Im Gegensatz zu dem BASIC-Schlüsselwort INPUT und GET bleibt das Programm bei DATA nicht stehen, sondern sucht sich die jeweiligen Daten selbst aus dem Programm aus. Dies geschieht über das Schlüsselwort READ.

4.

```
10 PRINT TAB(5)"ORT";SPC(5)"MITTEL-
                              TIEFSTEMPERATUR"
20 FOR I=1 TO 2
30 READ A$,B,C
40 PRINT TAB(5)A$,B,C
50 NEXT I
60 DATA MAINZ,15.2,-28,WIESBADEN,15.4,-27
70 RESTORE

READY.
```

Lösungen zu Kapitel 7:

1. Lösung siehe Seite 81.

2. Der Begriff Fallabfrage beinhaltet, daß ein Benutzer eines Programms unterschiedlich mögliche Abläufe in einem Programm auswählen kann. Fallabfrage heißt also immer Auswahlmöglichkeit.

3. Die Unterprogrammtechnik wird immer dann benutzt, wenn ein bestimmtes Programm im Laufe des Abarbeitens des Hauptprogramms mehrfach benutzt werden soll. Durch das Unterprogramm erspart man sich die mehrfache Codierung dieses Programmes; mit dem Schlüsselwort GOSUB wird dieses Unterprogramm angesprochen, wobei man im Unterprogramm als letzte Zeile den BASIC-Befehl RETURN schreiben muß, damit man aus dem Unterprogramm in diejenige Zeile zurückspring, die hinter der Zeile steht, in der man das Hauptprogramm verlassen hat. Ein Beispiel für ein Unterprogramm kann eine Rundungsroutine sein.

4. Der Grundgedanke der kaufmännischen Rundung ist folgender:

Beispielzahl	Operation	Wirkungsweise
100.238		
10023.8	*100	Das Komma wird um 2 Stellen nach rechts verschoben
10024.3	+0.5	Bei Zahlen von 5 und größer ergibt sich eine Zahl über 10, d.h. daß die Zahl vor dem Komma um 1 heraufgesetzt wird
10024	INT	Ganzzahlvariable, d.h. die Zahl nach dem Komma fällt weg
100.24	/100	Das Komma wird um 2 Stellen nach links verschoben

Die Codierung der Rundungsroutine entnehmen Sie bitte dem Beispiel Einfache Zinsrechnung in den Zeilen 1000 und 1020 auf Seite 90.

5. In den Zeilen 233 und 333 wird sichergestellt, daß das Anfangskapital, die Laufzeit und der Zinssatz für unsere Berechnung jeweils positiv sind. In der Wirtschaft ist es nicht möglich, ein negatives Kapital, negative Laufzeit und negativen Zinssatz zu haben. Der Computer kann aber selbst nicht erkennen, ob es sinnvoll ist, mit negativen Zahlen zu arbeiten, d.h. wir müssen als Programmierer dafür sorgen, daß in Fällen positiver Zahlen auch nur positive Zahlen eingegeben werden können. Die Zeilen 233 und 333 sind also Prüfzahlen für die logisch richtige Eingabe von positiven Zahlen.

6. Beide BASIC-Befehle haben dieselbe Wirkung; es wird der Bildschirm freigemacht, so daß die Ausgabe auf völlig geleertem Bildschirm erfolgt.

```
5 SCNCLR:PRINT
10 PRINT "PROZENTRECHNUNG"
20 PRINT:PRINT
30 :
40 REM AUSWAHLROUTINE
50 PRINT "WAS SOLL BERECHNET WERDEN?"
60 PRINT "BITTE AUSWAEHLEN!"
70 PRINT:PRINT
80 PRINT "PROZENTWERT   = 1 ":PRINT
90 PRINT "GRUNDWERT     = 2 ":PRINT
100 PRINT "PROZENTSATZ   = 3 ":PRINT
110 PRINT:PRINT
120 INPUT "IHRE GEWAEHLTE ZAHL =";A
130 SCNCLR
140 ON A GOTO 200,300,400
200 :
210 REM PROZENTWERTBERECHNUNG
220 INPUT"GRUNDWERT   =";GW
230 INPUT"PROZENTSATZ =";PS
235 IF PS<0 THEN PRINT"FALSCH":GOTO230
240 PW=GW*PS/100
250 PRINT:PRINT
260 PRINT "DER PROZENTWERT =";PW
270 GOSUB 500
300 :
310 REM GRUNDWERTBERECHNUNG
320 INPUT"PROZENTSATZ =";PS
325 IF PS<0 THEN PRINT"FALSCH":GOTO320
330 INPUT"PROZENTWERT =";PW
340 GW=PW*100/PS
350 PRINT:PRINT
360 PRINT "DER GRUNDWERT =";GW
370 GOSUB 500
400 :
410 REM PROZENTSATZBERECHNUNG
420 INPUT"GRUNDWERT   =";GW
430 INPUT"PROZENTWERT =";PW
440 PS=PW*100/GW
450 PRINT:PRINT
460 PRINT "DER PROZENTSATZ =";PS
470 :
500 INPUT "NOCH EINE BERECHNUNG?J/N";B$
510 IF B$="J" THEN 40
520 IF B$<>"J" THEN END

READY.
```

Lösungen zu Kapitel 8:

1. Es werden 11 Speicherplätze reserviert (siehe Seite 94).

2. Lösung siehe Seite 94 ff.

3. Die FOR...TO-Schleife ist im Zusammenhang mit der Tabellenverarbeitung deswegen sehr nützlich, weil man den Schleifenzähler I zugleich als Index für die jeweiligen Felder benutzen kann. Der Schleifenzähler wird um eins heraufgesetzt; dadurch kann man I als diejenige Variable benutzen, die die Felder in der entsprechenden Dimensionierung anspricht. Vergleiche hierzu Seite 97.

4. Lösung siehe Seite 99.

5.

```
5 SCNCLR:PRINT
10 PRINT "ANSCHRIFTEN"
20 PRINT:PRINT
30 REM DIE ANSCHRIFTEN WERDEN GETRENNT
31 REM NACH NAME / STRASSE / POSTLEIT-
32 REM ZAHL / WOHNORT EINGEGEBEN
33 REM BIS ZU 100 SIND HIER MOEGLICH
34 REM IN VERBINDUNG MIT EINEM SORT-
35 REM PROGRAMM KANN NACH JEDEM
36 REM KRITERIUM AUSGEGEBEN WERDEN!
40 :
50 REM VARIABLE SIND N$=NAME S$=STRASSE
51 REM P$=PLZ W$=WOHNORT
60 :
70 DIM N$(100),S$(100),P$(100),W$(100)
80 :
90 REM EINGABEROUTINE
100 FOR I=1 TO 100
110 INPUT "NAME    ";N$(I)
120 INPUT "STRASSE ";S$(I)
130 INPUT "PLZ     ";P$(I)
140 INPUT "WOHNORT ";W$(I)
150 PRINT
160 INPUT"NOCH EINE ANSCHRIFT?J/N";A$
170 IF A$="J" THEN 190
180 IF A$<>"J" THEN 200
190 NEXT I
200 :
210 REM AUSGABEROUTINE
220 FOR I=1 TO 100
230 SCNCLR
240 PRINT TAB(10) N$(I)
250 PRINT TAB(10) S$(I)
260 PRINT TAB(10) P$(I)
270 PRINT TAB(10) W$(I)
275 PRINT:PRINT
280 INPUT "NAECHSTE ANSCHRIFT?J/N";B$
290 IF B$="J" THEN 310
300 IF B$<>"J" THEN 320
310 NEXT I
320 END

READY.
```

Lösungen zu Kapitel 9:

1. Lösung siehe Seite 108.

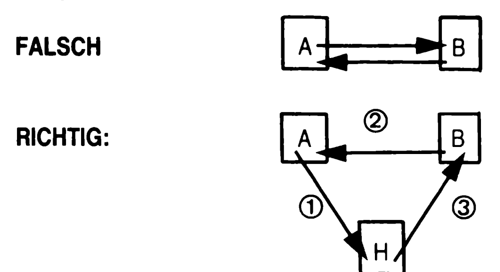
2. Wenn das Programm nicht mehr abläuft und die Fehlermeldung einen SYNTAX-ERROR ergibt, so kann durch einfaches Drücken der Taste HELP diejenige Anweisung gefunden werden, die den Fehler enthält. Geben Sie folgendes kleines Programm ein und prüfen Sie die Wirkung:

   10 PRINNT

   Das Gerät meldet einen SYNTAX-Fehler in Zeile 10. Wir drücken die Taste HELP und das falsch geschriebene Wort PRINNT blinkt auf dem Bildschirm.

3. Sortieren bedeutet, daß die niedrigsten Zahlen an den Anfang einer Reihe kommen und die höchsten Zahlen an das Ende dieser Reihen. Dazu ist es notwendig, daß wir immer zwei Zahlen miteinander vergleichen und, wenn die letzte Zahl kleiner ist als die erste, die Plätze ausgetauscht werden.

4. Platztausch bedeutet, daß der Inhalt der ersten Variable in den Inhalt der zweiten Variablen übernommen wird; umgekehrt der Inhalt der zweiten Variable soll in den Inhalt der ersten Variablen übergehen.



Für den Platzaustausch brauchen wir also immer noch ein zusätzliches 3. Feld, in das wir den Inhalt einer Variablen speichern, denn beim Übertragen von Werten wird automatisch der Inhalt von Variablen überschrieben.

5.

```
5 SCNCLR :PRINT
10 PRINT "RIPPLE - SORT"
20 PRINT:PRINT
30 INPUT "WIEVIELE WERTE SIND ZU SORTIEREN?";N
40 DIM A(N)
50 :
60 REM EINGABEROUTINE
70 PRINT:PRINT
80 FOR I=1 TO N
90 PRINT I". WERT =";:INPUT A(I)
110 NEXT I
120 :
130 REM ENDBEDINGUNG
140 V=0
150 :
160 REM SORTIERROUTINE
170 :
180 FOR I=1 TO N-1
190 IF A(I)<=A(I+1) THEN 220
200 H=A(I):A(I)=A(I+1):A(I+1)=H
210 V=1
220 NEXT I
230 :
240 REM ENDABFRAGE
250 IF V=1 THEN 140
260 :
270 REM AUSGABEROUTINE
280 PRINT:PRINT
290 FOR I=1 TO N
300 PRINT A(I)
310 NEXT I
320 END

READY.
```

6. Das Sortierprogramm muß von numerischen Variablen auf alphanumerische Variable umgestellt werden. Dies bedeutet, daß wir an die Variablen ein $-Zeichen anfügen müssen. Dieses $-Zeichen muß an die Variable der Zeile 40, 90, 190, 200 und 300. Beachten Sie, daß auch das 3. Feld H des Platzaustausch mit dem $-Zeichen versehen werden muß.

7. TRON RETURN

READY

C

RUN RETURN

[5]

[10] RIPPLE-SORT

[20]

[20]

[30] WIEVIELE WERTE SIND ZU SORTIEREN ??

[40] [60] [70]

[70]

[80] [90] 1. WERT = [90] ?

[110] [90] 2. WERT = [90] ?

[110] [90] 3. WERT = [90] ?

[110] [130] [140] [160] [180] [190] [200] [200]

[200] [210] [220] [190] [220] [240] [250] [140]

[160] [180] [190] [220] [190] [220] [240] [250]

[270] [280]

[280]

[290] [300]

[310] [300]

[310] [300]

[310] [320]

READY

C

TROFF RETURN

READY

C

Lösungen zu Kapitel 10:

1. Lösung siehe Seite 120 ff.

2. Jede Diskette ist fabrikneu nicht für den Computer zu gebrauchen. Sie muß erst für den Computer zurechtgemacht werden. Dies geschieht über das BASIC-Wort HEADER.

3. Lösung siehe Seite 123 f.

4. Lösung siehe Seite 124 f.

ANHANG 2: Zusammenfassung der wichtigsten BASIC-Schlüsselworte und deren Wirkung

1. Elementare Anweisungen und Kommandos

Anweisung	Wirkung (Beispiel)
AUTO	Automatische Zeilennumerierung AUTO 10 Programm neu numerieren: 10, 20, 30 ...
CLR	Löscht alle Variablenwerte
CONT	Ausführung fortsetzen mit Zeile der Unterbrechung
COLOR	Einfärben von geschlossenen Flächen 10 COLOR Farbzonen-#, Farb-#(Helligkeit)

Farbzonen-#
- 0 = Hintergrund des Bildschirms
- 1 = Vordergrund (Zeichen)
- 2 = Mehrfarben 1
- 3 = Mehrfarben 2
- 4 = Rahmen bzw. Rand des Bildschirms

Farb-#

1 = Schwarz	9 = Orange
2 = Weiß	10 = Braun
3 = Rot	11 = Gelbgrün
4 = Zyan	12 = Rosarot
5 = Violett	13 = Blaugrün
6 = Grün	14 = Hellblau
7 = Blau	15 = Dunkelblau
8 = Gelb	16 = Hellgrün

Helligkeit Von 0 = dunkel bis 7 = strahlend hell.

DATA	Daten im Programm speichern: 10 DATA 11,"DM" Daten programmintern speichern und 11 READ A,AS	
DEF FN ...	Definieren einer Funktion	
DELETE	Zeilen des im RAM befindlichen Programmes löschen	
DIM	Dimensionieren von Feldern: 10 DIM A (2,2) Feld A mit 3 Zeilen / 3 Spalten	
DO-LOOP-UNTIL	Nicht-abweisende Schleife: 10 DO 20 LOOP UNTIL Z = 10	 Wiederhole die Anweisungen zwischen 10 und 20, bis (= until) die Bedingung Z = 10 erfüllt ist.
DO-WHILE-LOOP	Abweisende Schleife: 10 DO WHILE Z=0 ... 20 LOOP	 Wiederhole die Anweisungen zwischen 10 und 20, solange (= while) die Bedingung Z=0 erfüllt ist.
DRAW	Linien und Punkte zeichnen: 10 DRAW (Farbzonen-#), (X,Y) (T0, X1, Y1) Zonen-# Farbzonen 0-3 X, Y Anfangspunkt (Standard = Cursorposition) X, Y Endpunkt (Standard = X1, Y1, d.h. Punkt).	
END	Beenden der Programmausführung	
FOR NEXT	Zählerschleife: 10 FOR I = 1 TO 10 20 PRINT I 30 NEXT I	 Zählerschleife gibt Werte 1, 2 ... 10 der Laufvariablen I aus.
FRE (0)	Für Anwender verfügbaren Speicherplatz zeigen FRE (0). Im RAM frei verfügbar z.B. 12275	
GET	Einzelnes Zeichen von Tastatur lesen	
GETKEY	Einzelnes Zeichen von Tastatur lesen	
GOSUB - RETURN	Unterprogrammsteuerung 100 GOSUB 100 20 ...	 Unterprogramm ab Zeile 100 aufrufen, ausführen und mit RETURN nach Folgezeile 20.
GOTO	Unbedingte Verzweigung 10 GOTO 50	 Von Zeile 10 unbedingt zu 50 verzweigen.

KEY	Belegung der Funktionstasten anzeigen
IF - THEN	Verzweigung nach Entscheidung: einseitige Auswahl 10 IF A=3 GOTO 50 Wenn A=3, dann nach 50 verzweigen. 10 IF A=3 GOTO 50 Verzweigung wie mit GOTO. 10 IF AS="JA" THEN PRINT "Richtig" (d.h. Ausgabe für Fall "JA").
IF-THEN-ELSE	Verzweigung nach Entscheidung: zweiseitige Auswahl 10 IF A=3 THEN 50:ELSE Wenn A=3, dann nach 50 verzweigen, sonst nach 60.
INPUT	Eingabe über Tastatur 10 INPUT A Eingabewert A zuweisen. 10 INPUT "Welche Zahl": A Eingabekommentar zusätzlich. 10 INPUT WS STRING als Eingabe
INT	Ganzzahliger (integer) Teil von Zahl A 10 PRINT INT (12.34) Ganzzahliger Teil von 12.34 ist 12
LET	Wertzuweisung 10 LET A=1 Wert 5 der Variablen A zuweisen. 10 LET A=A-1 Wert von A um 1 erhöhen. 10 LET Z=K*P*T (100*360) Wert berechnen und Z zuweisen.
LIST	Auflisten der BASIC-Codierung LIST Alle Zeilen des Programms auflisten LIST 10 Nur die Zeile 10 auflisten. LIST 10- LIST -100 LIST -100-300 Listen von, bis, von-bis.
MONITOR	Aufruf des eingebauten Maschinensprache-Monitors
NEW	Löschen des Hauptspeichers NEW Im RAM befindliches Programm und alle Variablen löschen.
ON-GOSUB	Fallabfrage mit Unterprogrammaufruf 10 ON GOSUB 100. 200. 300 für G=1 nach 100, für G=2 nach 200 und für G=3 nach 300 verzweigen.
ON-GOTO	Fallabfrage mit Verzweigung 10 ON G GOTO 100. 200. 300 für G=1 nach 100 verzweigen, für E=2 nach 200, für E=3 nach 300, für E=0 nach Folgezeile.
PEEK	Speicherplatz direkt lesen

POKE	Speicherplatz direkt beschreiben	
PRINT	Ausgabe auf Bildschirm	
	10 PRINT A	Werte von Variable A ausgeben.
	10 PRINT A,"DM"	Wert der Variable A und Text "DM" ausgeben.
PRINT USING	Formatierte Ausgabe mittels Formatfeld bzw. string:	
	10 MS=###.##	MS als Druckmaske
	20 PRINT USING MS;123456	Konstante 123456 und Wert von A
	30 PRINT USING MS;A	formatiert ausgeben.
PUDEF	Eines der 4 Zeichen ",.S" im Formatstring neu definieren	
READ	Lesen von Daten aus DATA-Zeile:	
	10 READ A	Nächsten Wert aus DATA nach A einlesen.
REM	Bemerkungen in BASIC-Codierungen einfügen	
	10 REM ÜBERSCHRIFT	Bemerkung bei LIST zeigen, nicht aber bei RUN.
RENUMBER	Zeilennummer des im RAM stehenden Programms numerieren	
RESTORE	Lesezeiger auf Position 1 zurücksetzen. Pointer der DATA-Zeile wird auf den 1. Wert zurückgesetzt, um dieselben DATA-Werte erneut lesen zu können.	
RESUME	Nach Fehlerbehandlung mit TRAP Ausführung fortsetzen.	
RUN	Ausführen eines Programms im Hauptspeicher	
	RUN	Das gerade im RAM befindliche Programm ausführen.
SCNCLR	Bildschirm löschen und Cursor nach links oben bringen	
	10 SCNCLR	Bildschirm frei und Cursor nach links oben.
STOP	Abbrechen der Programmausführung	
SYS	Sprung in ein Maschinenprogramm	
TAB(X)	Tabulator-Funktion zur Ausgabe	
	10 PRINT TAB(10);"A"	"A" wird in Spalte 10 ausgegeben.
TRAP	Zu einer Fehlerbehandlungsroutine verzweigen	
TRON-TROFF	Einen Trace-Lauf beginnen bzw. beenden:	
	TRON	Nach RUN das Programm schrittweise ausführen.
	TROFF	Trace-Modus wieder ausschalten.
WAIT	Warten, bis eine angegebene Speicherstelle einen bestimmten Wert hat.	

2. Ausgewählte Anweisungen und Kommandos für Peripheriegeräte

CMD	Datenkanal offenhalten: z.B. Drucken statt zum Monitor. OPEN 1.4: CMD1:LIST PRINT#1 CLOSE Kanal 1 zum Drucker (Gerät 4) öffnen, Programm listen, Kanal schließen.	
DIRECTORY	Disketten-Inhaltsverzeichnis zeigen.	
DLOAD	Laden eines Programms von Diskette in den RAM	
	DLOAD "ZINS"	RAM löschen und eine Kopie des Programm ZINS von Diskette in die ZE bringen.
	DLOAD"ZINS",D1	Von Diskettenlaufwerk 1 laden.
	10 DLOAD"ZINS"	ZINS laden und starten.
DSAVE	Speichern bzw. Retten eines Programms auf Diskette	
	DSAVE"ZINS"	Eine Kopie des gesamten RAM-Inhaltes unter dem Namen ZINS auf Diskette erstmalig speichern.
HEADER	Anlegen einer neuen Diskette	
	HEADER"FH".D0.I01	Diskette FH in D0 (Identifikation 01). Name und Identifikation beibehalten.
LOAD	Laden eines Programms von Datasette in die ZE	
	LOAD"ZINS"	Hauptspeicher RAM löschen und eine Kopie von Programm ZINS in die ZE bringen.
OPEN	Öffnen einer logischen Datei zur Vorbereitung des Datenverkehrs mit einem externen Gerät:	
	10 OPEN 1.4	Drucker mit Gerätenummer 4.
RUN	Programm in den RAM laden und ausführen	
	RUN"ZINS"	Derzeitigen RAM-Inhalt löschen, Programm ZINS von Diskette laden und ZINS dann ausführen
SAVE	Speichern eines Programms auf Kassette	
	SAVE"ZINS"	Inhalt des RAM unter dem Programmnamen ZINS auf Datasette speichern.
SCRATCH	Löschen einer Datei auf Diskette	
VERIFY	Vergleichen eines Programmes im RAM und auf Kassette	
	VERIFY"ZINS"	Kontrollvergleich des Programms ZINS.

ANHANG 3: Fehlermeldungen

Die folgenden Fehlermeldungen werden durch das BASIC V3.5 ausgegeben, wenn im Verlaufe des Programms auf sie gestoßen wird.

Fehler-Name:	**Fehler-Quelle:**
BAD DISK	Die eingelegte Diskette ist kaputt oder sie war nicht formatiert.
BAD SUBSCRIPT	Die DIM-Anweisung ist nicht korrekt: das angesprochene Arrey ist nicht vorhanden.
BREAK	Der STOP Befehl wurde ausgeführt.
CAN'T CONTINUE	Das Programm lief nicht: das CONT Kommando kann nicht ausgeführt werden.
CAN'T RESUME	Der RESUME-Befehl steht ohne TRAP-Befehl.
DEVICE NOT PRESENT	Das geforderte Eingabe Ausgabe Gerät steht nicht bereit.
DIRECT MODE ONLY	Das Kommando kann nur im Direkt-Modus benutzt werden, nicht im Programm.
DIVISION BY ZERO	Eine Variable hat den Wert Null: durch Null kann nicht geteilt werden.
FILE DATA	Falsche Daten gelesen.
FILE OPEN	Die benutzte FILE ist bereits geöffnet.
FILE NOT OPEN	Die bentutzte FILE ist noch nicht geöffnet.
FILE NOT FOUND	Der FILE-Name (Nummer) wurde noch nicht vergèben.
FORMULA TOO COMPLEX	Die benutzte Formel (Ausdruck) ist zu komplex und muß aufgeteilt werden.
ILLEGAL DEVICE NUMBER	Die Nummer des angesprochenen Peripherie-Gerätes ist falsch.

ILLEGAL DIRECT	INPUT oder GET Befehle sind nur innerhalb eines Programms erlaubt.
ILLEGAL QUANTITY	Eine Zahl ist außerhalb des erlaubten Bereichs.
LOAD	Der LOAD-Befehl funktionierte nicht: nochmals versuchen.
LOOP NOT FOUND	Im Programm ist ein D0-Befehl: der zugehörige LOOP-Befehl kann nicht gefunden werden.
LOOP WITHOUT DO	Im Programm ist ein LOOP-Befehl; der zugehörige D0-Befehl kann nicht gefunden werden.
MISSING FILE NAME	Ein OPEN. LOAD oder SAVE-Befehl zur Diskette erfordert unbedingt einen FILE-Namen.
NEXT WITHOUT FOR	Schleifenfehler: der NEXT-Befehl stimmt mit dem FOR .. TO Befehl nicht überein.
NOT INPUT FILE	Es wurde der Versuch unternommen. einen INPUT oder GET-Befehl auf ein. als Ausgabe festgelegte Einheit, anzuwenden.
NOT OUTPUT FILE	Es wurde der Versuch gemacht. auf eine Eingabe-Einheit Daten auszugeben.
NO GRAPHICS AREA	Der Befehl DRAW etc. steht zu früh: es muß vorher der GRAPHIC-Befehl erfolgen.
OUT OF DATA	Ein READ-Befehl wurde benutzt. obwohl alle Daten bereits abgearbeitet sind.
OUT OF MEMORY	Der Speicher ist gefüllt: es ist kein Platz mehr für das Programm oder Programmvariable oder es wurden zuviele DO. FOR oder GOSUB-Befehle benutzt.
OVERFLOW	Das Ergebnis einer Berechnung ist größer als die größte zugelassene Zahl.
REDIM'D ARREY	Ein Arrey kann nur einmal dimensioniert werden.
RETURN WITHOUT GOSUB	Ein RETURN-Befehl wurde benutzt, obwohl ein GOSUB-Befehl nicht vorhanden ist.
STRING TOO LONG	Ein STRING kann lediglich bis einschließlich 255 Zeichen beinhalten.

SYNTAX	Der benutzte BASIC-Befehl ist nicht korrekt. Der Fehler kann ein Schreibfehler. fehlen von Anführungsstrichen, Semikolon statt Komma etc. sein.
TOO MANY FILES	Das System kann lediglich bis einschließlich 10 geöffnete FILES verwalten; es sind mehr als 10 geöffnet worden.
TYPE MISMATCH	Eine numerische Variable wurde anstelle einer alphanumerischen Variablen genutzt oder umgekehrt.
UNDEF'D FUNKTION	Die angesprochene Funktion ist noch nicht definiert worden.
UNDEF'D STATEMENT	Die angesprochene Zeilennummer existiert nicht im Programm.
VERIFY	Das Programm auf Kassette oder Diskette stimmt mit dem Programm in der ZE nicht überein.

DIE COMMODORE SACHBUCHREIHE

Band 1:
ALLES ÜBER DEN COMMODORE 64
ISBN 3-89 133-000-6
(Artikel-Nr. 55 64 20)
1984, 480 S., Fe. Ebd.
DM 59,–

Band 2:
ALLES ÜBER DEN VC 20
ISBN 3-89 133-004-9
(Artikel-Nr. 58 00 20)
1984, 200 S., Br.
□ **DM 9,80**

Band 3:
LOGO FÜR COMMODORE
mit 2 Disketten
ISBN 3-89 133-001-4
(Artikel-Nr. 64 10 50)
1984, 364 S., A4, Fe. Ebd.
DM 159,–

Band 4:
DAS COMMODORE 64
SPIELE-BUCH
ISBN 3-89 133-002-2
(Artikel-Nr. 55 64 15)
1984, 160 S., Br.
DM 29,80

Band 5:
DAS VC 20 SPIELE-BUCH
ISBN 3-89 133-003-0
(Artikel-Nr. 58 00 15)
1984, 160 S., Br.
DM 29,80

Band 6:
PLUS/4 ROM-LISTING
ISBN 3-89 133-006-5
(Artikel-Nr. 58 40 00)
1984, 280 S., Br.
DM 59,–

Band 7:
AUTOMATEN UND SENSOREN
ZUM SELBERBAUEN FÜR
COMMODORE COMPUTER
von John Billingsley
ISBN 3-89 133-007-3
(Artikel-Nr. 58 00 05)
1984, 128 S., Br.
DM 24,80

Band 8:
MATHEMATIK MIT DEM
COMMODORE 64
von Czes Kosniowski
mit 1 Diskette
ISBN 3-89 133-008-1
(Artikel-Nr. 55 64 30)
1984, 166 S., Br.
DM 34,80

Band 9:
PROGRAMMIERTECHNIKEN FÜR
FORTGESCHRITTENE AUF DEM
COMMODORE 64
von David Lawrence
ISBN 3-89 133-009-X
(Artikel-Nr. 55 64 35)
1985, 184 S., Br.
DM 29,80

Band 10:
DER COMMODORE 64 ALS
GRAFIK-KÜNSTLER
von Boris Allan
ISBN 3-89 133-010-3
(Artikel-Nr. 55 64 38)
1985, 144 S., Br
DM 24,80

Preisänderungen vorbehalten
□ Unverbindliche Preisempfehlung

Band 11:
DER COMMODORE 64
IN DER PRAXIS
von David Lawrence
ISBN 3-89 133-011-1
(Artikel-Nr. 55 64 23)
1985, 200 S., Br. **DM 29,80**

Band 12:
PROGRAMMIERUNG IN
MASCHINENSPRACHE AUF DEM
COMMODORE 64: DAS WERKZEUG
von Mark England und
David Lawrence
ISBN 3-89 133-012-X
(Artikel-Nr. 55 64 25)
1985, 208 S., Br. **DM 29,80**

Band 13:
PROGRAMMIERUNG IN
MASCHINENSPRACHE AUF DEM
COMMODORE 64:
GRAFIK UND MUSIK
von Mark England und
David Lawrence
ISBN 3-89 133-013-8
(Artikel-Nr. 55 64 26)
1985, 248 S., Br. **DM 29,80**

Band 14:
PROGRAMMIERUNG IN
MASCHINENSPRACHE AUF DEM
COMMODORE 64: SPIELE
von Paul Roper
ISBN 3-89 133-014-6
(Artikel-Nr. 55 64 27)
1985, 192 S., Br. **DM 29,80**

Band 15:
KÜNSTLICHE INTELLIGENZ AUF
DEM COMMODORE 64
von Keith und Steven Brain
ISBN 3-89 133-015-4
(Artikel-Nr. 55 64 31)
1985, 160 S., Br. **DM 29,80**

Band 16:
FLOPPY-PROGRAMMIERUNG
MIT DEM COMMODORE 64
von David Lawrence und
Mark England
ISBN 3-89 133-016-2
(Artikel-Nr. 55 64 28)
1985, 168 S., Br. **DM 29,80**

Band 17:
STRATEGIESPIELE AUF DEM
COMMODORE 64
EINE PROGRAMMIERANLEITUNG
ISBN 3-89 133-017-0
(Artikel-Nr. 55 64 32)
1985, 152 S., Br. **DM 29,80**

Band 18:
DER COMMODORE 16
IN DER PRAXIS
ISBN 3-89 133-018-9
(Artikel-Nr. 55 50 00)
1985, 208 S., Br. **DM 29,80**

Band 19:
ALLES ÜBER DEN
COMMODORE 128
ISBN 3-89 133-019-7
(Artikel-Nr. 55 64 81)
in Vorbereitung (Oktober '85)

Band 20:
DAS C128-BASIC-BUCH
von Hans-Lorenz Schneider
ISBN 3-89 133-20-0
(Artikel-Nr. 55 64 82)
in Vorbereitung (Oktober '85)

Band 21:
DAS C128-BUCH
FÜR UMSTEIGER
von Hans-Lorenz Schneider
ISBN 3-89 133-021-9
(Artikel-Nr. 55 64 83)
in Vorbereitung (Oktober '85)

Sachwortverzeichnis:



Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Langenhagstraße 1
CH-4147 Aesch

Commodore GmbH
Kinskygasse 40-44
A-1232 Wien

Artikel-Nr. 580117/2.86 Änderungen vorbehalten.

